SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Ime Prezime:
Tin Mihaljevski Boltek

Zagreb, 2019.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor:
Prof.Dr.Sc. Josip Kasać

Ime Prezime:
Tin Mihaljevski Boltek

Zagreb, 2019.

Izjavljujem da sam ovaj rad izradio samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se svome mentoru, prof.dr.sc. Josipu Kasaću na pruženoj potpori, podršci i savjetima tIjekom izrade ovog rada.
Zahvaljujem se i kolegama Denisu Kotarskom i Petru Piljeku na podršci i svim savjetima.
Također zahvaljujem se svojoj obitelji, majci Štefaniji, ocu Slobodanu i sestri Lei i prijateljima na moralnoj podršci tijekom cijelog studija, kao i djevojci Andrei na koja me trpila kroz ova teška vremena.

Tin Mihaljevski Bolte

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne ispite studija strojarstva za smjerove:
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa:	
Ur.broj:	

ZAVRŠNI ZADATAK

Student: **TIN MIHALJEVSKI-BOLTEK** Mat. br.: 0035200549

Naslov rada na hrvatskom jeziku: **DALJINSKO UPRAVLJANJE LETJELICOM S ČETIRI ROTORA**

Naslov rada na engleskom jeziku: **REMOTE CONTROL OF QUADCOPTER**

Opis zadatka:

Brzi razvoj sve jeftinije elektronike, od senzora pa do mikroprocesora, omogućuje sve širu i dostupniju primjenu bespilotnih letjelica. Daljinsko upravljanje bespilotnim letjelicama zahtijevan je zadatak s obzirom da je dinamika letjelice opisana nelinearnim multivarijabilnim dinamičkim modelom koji uključuje razne neodređenosti, od nepoznatih parametara do nemjerljivih vanjskih poremećaja. Osnovni zadatak ovog rada je sinteza i implementacija daljinskog upravljanja letjelicom s četiri rotora u slučaju prenošenja tereta nepoznatih masa i momenata inercija.

U radu je potrebno:

- Za izrađeni fizički model letjelice s četiri rotora ugraditi odgovarajuće elektroničke sklopove za prikupljanje i obradu podataka sa senzora, te za upravljanje elektromotorima.
- Implementirati biblioteke programa za upravljanje elektroničkim sklopovima letjelice.
- Implementirati nelinearni i linearizirani dinamički model letjelice u programskom paketu Matlab.
- Provesti sintezu regulatora na temelju lineariziranog modela letjelice.
- Testirati upravljačke algoritme za različite manevre i različite mase letjelice.

Zadatak zadan:
29. studenog 2018.

Rok predaje rada:
1. rok: 22. veljače 2019.
2. rok (izvanredni): 28. lipnja 2019.
3. rok: 20. rujna 2019.

Predviđeni datumi obrane:
1. rok: 25.2. - 1.3. 2019.
2. rok (izvanredni): 2.7. 2019.
3. rok: 23.9. - 27.9. 2019.

Zadatak zadao:

Kasać Josip
Prof. dr. sc. Josip Kasać

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS OZNAKA

Oznaka	Jedinica	Opis
Γ	-	Vektor pozicije u inercijskom okviru
Θ	-	Vektor orijentacije u inercijalnom okviru
x	m	Pozicija u smjeru X osi
y	m	Pozicija u smjeru Y osi
z	m	Pozicija u smjeru Z osi
Φ	°	Kut oko X osi
θ	°	Kut oko Y osi
ψ	°	Kut oko Z osi
u	m/s	Brzina u smjeru X osi referentnog sustava vezanog uz tijelo
v	m/s	Brzina u smjeru Y osi referentnog sustava vezanog uz tijelo
w	m/s	Brzina u smjeru Z osi referentnog sustava vezanog uz tijelo
$\mathbf{R}$	-	Matrica rotacija
$\mathbf{\Omega}_B$	-	Matrica transformacija
$\mathbf{I}$	-	Diag matrica inercija
$\boldsymbol{\tau}$	-	Vektor momenata motora
$\mathbf{d}_{\boldsymbol{\tau}}$	-	Vektor poremećaja
m	kg	masa
$\mathbf{F}$		Vektor vanjskih sila
$\mathbf{D}_{\mathbf{F}}$		Vektor vanjskih sila poremečaja
l	m	duljina
c	m	Konstanta moment-sila
e	-	Regulacijska pogreška

SAŽETAK

Ovaj rad bavi se implementacijom elektroničkog sklopovlja za daljinsko upravljanje kvadkopterom. Bilo je potrebno povezati senzore poput akcelerometra i žiriskopa, radio prijemnik, te elektromotore s kontrolerom. Napisan je programski kod za kontroler koji omogućuje upravljanje elektromotorima letjelice preko radio odašiljača kojim upravlja operater. Dinamički model letjelice s regulatorom brzine implementiran je i testiran u Matlabu.

Ključne riječi: kinematika i dinamika krutog tijela, kvadkopter, linearno upravljanje

SUMMARY

This paper deals with the implementation of electronics for remote controled drone. It was necessery to connect sensors such as accelerometer and gyroskope, radio receiver and electric motors with a controller. A contoller code has beed written to control the electric motors via and operator-controlled radio transmiter. A dynamic model of the aircraft with a speed controller was implemented and tested in Matlab.

Key words: kinematic and dynamics of rigid bodies, quadcopter, linear control

1. UVOD

Ovaj rad se bavi softverskim aspektima kvadkopter helikoptera. Iako su prvi prototipovi izrađeni još početkom dvadesetog stoljeća, kvadkopter i zajedno sa čitavim područjem letećih mobilnih robota doživljavaju brz razvoj i širenje primjene tek u posljednjih desetak godina. Razloge za to nalazimo u munjevitom razvoju mikroprocesora, koji se ponajprije očituje u povećanju njihove procesorske moći te u minijaturizaciji pripadnih kućišta, zatim u razvoju naprednih električnih sklopova i preciznih senzorskih modula malih dimenzija koji su nužni za stabilizaciju inherentno nestabilnih letjelica poput kvadkoptera. Razvoj baterija s visokim omjerom kapaciteta i mase pridonio je razvoju letjelica s duljim vremenom autonomije i većih dimenzija, čime se povećava spektar mogućih primjena. Vrlo su popularni u istraživačkim laboratorijima jer kao inherentno nestabilni podaktuirani nelinearni tehnički sustav sa šest stupnjeva slobode predstavljaju svestranu platformu za razvoj i testiranje novih upravljačkih algoritama. Kvadkopteri se danas koriste za fotografiju i snimanje, u filmskoj industriji, za dostavu paketa. Spektar mogućih primjena brzo se širi te se ubrzano usavršavaju njihove primjene u pametnoj agrikulturi, nadzoru zgrada i granica, inspekciji gradilišta, održavanju.

Slika 1. Razne namjene

Dron na kojemu se radi u ovom radu je rad kolegice Jelene Širjan. Dron forme +, gdje se krakovi drona poklapaju s x i y koordinatama referentnog koordinatnog sustava sa središtem u centru mase letjelice. Središte drona je napravljeno 3D printanjem, kao I nosači motora, dok su krakovi od karbonskih cijevi. Motori su SunnySky x2216 880Kv, s Gemfan 11x47 propelerima, koje pogone Turnigy Plush 30A ESC I sve je napajano 5000mAh LiPo baterijom kojom bi se trebalo dobiti vrijeme leta oko 10 minuta.
Sila potiska se ostvaruje vrtnjom motora, odnosno propelera koji funkcioniraju na istom principu kao i krilo aviona, svojom geometrijom mijenja brzinu gibanja fluida, a iz Bernulijeve jednadžbe znamo da ako se brzina strujanja smanji, tlak raste, ako dobijemo polje višeg tlaka na donjoj strain propelera dobijemo potisnu silu.

Slika 2. Dron

U radu je kratko obrađena kinematika i dinamika letjelice, kako je neophodna za sintezu regulatora. Sinteza regulatora za drona predviđenog za dostavu, tako da bi mu masa bila nepoznata pošto ne bismo nužno znali što prenosimo, upravljivog po poziciji. Te sinteza nešto jednostavnijeg regulatora za upravljanje po brzini, kojim bi se dronom upravljao ručno sa Zemlje pomoću joisticka. Dalje slijede rezultati simulacija u Matlabu I opis praktičnog dijela rada.

Praktični dio rada se zasniva na Raspberry Pi-u, malom računalo koje nije namijenjeno ovakvoj primjeni i nikako nije idealni izbor, ali zbog svoje rasprostranjenosti i cijene dali smo mu šansu.

2. Dinamički model

Dinamički model svakog realnog sustava je početak njegove kontrole. To je matematički zapis nečega stvarnoga, koji koristimo u danjoj obradi. Dinamički model je spoj kinematike i dinamike krutog tijela, stoga opisuje stanje letjelice u ovisnosti o silama aktuatora Kako dinamički model nije tema ovog završnog, biti će informativno pokazan, zato što kako je i prije rečeno nužan je za sintezu regulatora same letjelice, kao i svakog drugog sustava. Qvadrotor je podaktuirani sustav, zato što ga razmatramo kao kruto tijelo s šest stupnjeva slobode (eng. 6 DOF), a ima samo 4 aktuatora. Postoje 4 naredbe za upravljanje gibanja kvadkoptera:

- Potisak (eng. throttle) je naredba kojom se istovremeno i jednako povećava ili smanjuje brzina svih motora. Kako svi propeleri djeluju u istome smjeru stvaraju potisnu silu u smjeru osi z dolazi do dizanja ili spuštanja letjelice. Dok letjelica lebdi vertikalna potisna sila poništava djelovanje gravitacijske sile na tijelo.
- Poniranje (eng. pitch) je naredba kojom postižemo gibanje unaprijed ili unazad u horizontalnoj ravnini u smjeru osi x. Dron se rotira oko svoje y osi povećavanjem odnosno smanjivanjem brzine vrtnje prednjih i stražnjih motora. Potisna sila se rastavi na vertikalnu i horizontalnu koja mu daje brzinu u odgovarajućem smjeru.
- -Valjanje (eng. roll) je naredba u suštini ista kao i poniranje samo oko druge osi letjelice, dolazi do promjene brzine vrtnje lijevih i desnih motora i na istom principu, kako je i prije objašnjeno, dolazi do kretanja.
- -Skretanje (eng. yaw) je naredba koja uzrokuje zakretanje letjelice oko vertikalne osi, postiže se promjenom omjera brzine vrtnje na dijagonalama, kad se više momenti motora međusobno ne ponište i dolazi do zakreta zbog reaktivnog momenta oko vertikalne osi kvadkoptera.

Izvod dinamičkog modela može se vidjeti u [2]. Jedina je razlika u okviru, koja dolazi do izražaja samo u matrici alokacije sila. Za primjer drona na kojem je baziran ovaj rad forma okvira je +, tako da je matrica alokacije potisaka jednostavnija.

Kinematika je grana mehanike koja proučava gibanje tijela ili sustava tijela bez razmatranja uzroka tih gibanja. Bavi se opisom gibanja, pozicijama, brzinama i ubrzanjima. Sva gibanja su relativna stoga njihov opis ovisi o promatraču. Stoga da bismo opisali gibanja potrebno je strogo definirati u odnosu na što ih opisujemo. Referentni okvir je dogovoreni koordinatni sustav za koji opisujemo gibanja. Za opis složenih gibanja nije uvijek jednostavno i intuitivno opisati ih u referentnom okviru u kojem je to potrebno. U tom slučaju definiraju se drugi referentni okviri.

U našem slučaju definiraju se dva okvira , inercijski koji je vezan za Zemlju koji smatramo da ne rotira pošto utjecaj akceleracijskih efekata zbog rotacije Zemlje možemo zanemariti pošto su dovoljno mali. Također definiramo i drugi referentni okvir koji je kruto vezan za tijelo letjelice, kojemu se osi poklapaju s krakovima letjelice i središte je u centru mase letjelice.
Referentni okvir vezan za tijelo:

- matrica inercije u okviru tijela postaje vremenski nepromjenjiva,
- jednadžbe gibanja se pojednostavljuju zbog simetričnosti konstrukcije,
- vertikalna sila i moment koji zakreću letjelicu zadaju se u tom okviru,
- vertikalna sila u ovom okviru ne mijenja smjer,
- mjerenja sa senzora na letjelici odnose se na ovaj okvir.

Inercijalni okvir:

- definicija pozicije letjelice u prostoru ima smisla jedino u inercijskom (globalnom) okviru,
- inercijske sile u inercijskom okviru nestaju,
- gravitacijska sila postaje konstantna po smjeru i iznosu.

Iz pretpostavku krutog tijela, i teorema o krutom tijelu proizlazi da gibanje krutog tijela možemo promatrati kao translaciju i rotaciju referentnog okvira vezanog za letjelicu u odnosu na neki drugi, inercijalni (desnokretni, NED).
Pozicija letjelice se definira u inercijalnom okviru i definirana je s vektorom položaja, koji definira položaj ishodišta referentnog okvira letjelice. Orijentacija je opisana pomoću 3 Eulerova kuta Psi, Theta Psi.

$$\Gamma^I = [x_I y_I z_I]^T \qquad \Theta^I = [\Phi\theta\psi]^T \tag{2.1}$$

Gdje je:

- x_I pozicija središta referentnog okvira vezanog za letjelicu u smjeru osi X inercijalnog okvira
- y_I pozicija središta referentnog okvira vezanog za letjelicu u smjeru osi Y inercijalnog okvira
- z_I pozicija središta referentnog okvira vezanog za letjelicu u smjeru osi Z inercijalnog okvira
- Φ kut zakreta referentnog okvira vezanog za letjelicu oko X osi
- θ kut zakreta referentnog okvira vezanog za letjelicu oko Y osi
- ψ kut zakreta referentnog okvira vezanog za letjelicu oko Z osi

Opis orijentacije temelji se na Eulerovom rotacijskom teoremu, koji tvrdi da se svaki pomak krutog tijela tijekom kojeg neka točka tijela ostaje fiksna može opisati jednom rotacijom oko proizvoljne osi kroz tu točku. Vrijedi i obratno, svaku složenu rotaciju oko proizvoljne osi moguće je rastaviti na elementarne rotacije oko zadanih osi. Euler je dokazao da s tri kuta moguće prikazati sve rotacije, detaljnije u [3]. Takve rotacije opisujemo jednostavnim rotacijskim matricama koje opisuju rotaciju za određeni kut oko jedne osi. Množenjem tih rotacijskih matrica, definiranim redoslijedom rezultira rotacijskom matricom za transformaciju između inercijalnog i referentnog okvira tijela. Inverz te iste matrice možemo koristit za transformaciju iz referentnog okvira u inercijalni okvir, također vrijedi za transformacijske matrice da je inverz jednak transponiranoj. To će nam koristit kako bismo ubrzanje mjereno na letjelici prebacili u inercijalni okvir. Referentni okvir vezan za tijelo je također desnokretni s središtem u centru mase. U tom okviru definirani su vektori linearne i kutne brzine.

$$\boldsymbol{V^B} = [u\ v\ w]^{\boldsymbol{T}}$$
$$\omega^B = [p\ q\ r]^{\boldsymbol{T}} \qquad \textbf{(2.2)}$$

Gdje je:

- u brzina u smjeru X osi referentnog sustava vezanog uz tijelo letjelice,
- v brzina u smjeru Y osi referentnog sustava vezanog uz tijelo letjelice,
- w brzina u smjeru Z osi referentnog sustava vezanog uz tijelo letjelice,
- p kutna brzina oko osi X referentnog okvira vezanog za tijelo,
- q kutna brzina oko osi X referentnog okvira vezanog za tijelo,
- r kutna brzina oko osi X referentnog okvira vezanog za tijelo.

$$\dot{\boldsymbol{\Gamma}} = \mathbf{RV}. \qquad \textbf{(2.3)}$$

Gdje je :

- **R** matrica rotacija

$$\mathbf{R} = \mathbf{R}_z^T(\Psi)\mathbf{R}_y^T(\theta)\mathbf{R}_x^T(\Phi). \tag{2.4}$$

$$\boldsymbol{R}_x(\Phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\Phi & s_\Phi \\ 0 & -s_\Phi & c_\Phi \end{bmatrix}, \boldsymbol{R}_y(\theta) = \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix}, \quad \boldsymbol{R}_z(\psi) = \begin{bmatrix} c_\psi & s_\psi & 0 \\ -s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.5}$$

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{\Omega}_\mathbf{B}\boldsymbol{\omega}. \tag{2.6}$$

Gdje je:

- $\boldsymbol{\Omega}_\mathrm{B} = \begin{bmatrix} c_\theta & s_\Phi s_\theta & c_\Phi s_\theta \\ 0 & c_\Phi c_\theta & -s_\Phi c_\theta \\ 0 & s_\Phi & c_\Phi \end{bmatrix}$, matrica transformacija iz okvira na tijelu u inercijalni okvir, $c_n = \cos(n)$, $s_n = \sin(n)$.

Dinamika je grana mehanike koja povezuje promjene gibanja (pozicije, brzine i ubrzanja) tijela, s silama i momentima koji ih uzrokuju. Cilj su jednadžbe koje opisuju utjecaj sila na promjenu gibanja tijela. Uz dvije pretpostavke: ishodište referentnog okvira vezanog uz tijelo podudara se s centrom mase tijela i da se koordinatne osi referentnog okvira vezanog za tijelo podudaraju s glavnim osima krutog tijela.

Rotacijska dinamika triju osi krutog tijela u okviru vezanom uz tijelo je dana izrazom:

$$\mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) = \boldsymbol{\tau} + \boldsymbol{d}_\tau. \tag{2.7}$$

Gdje je:

- $\mathbf{I} = \mathrm{diag}\{I_x, I_y, I_z\}$, diagonalna matrica inercija,
- $\boldsymbol{\tau} = [\tau_\Phi \ \tau_\theta \ \tau_\psi \ ]^T$, vektor momenata motora,
- $\mathbf{d}_\tau = [d_\Phi \ d_\theta \ d_\psi]^T$, vektor poremećaja, vanjskih momenata.

$$\boldsymbol{m\dot{\omega} + \omega \times (mv) = F + d_F}. \quad \textbf{(2.8)}$$

Gdje je:

- m –masa,
- $\mathbf{F} = [F_x\, F_y\, F_z]^T$ vektor vanjskih sila,
- $\mathbf{D}_F = [d_x\, d_y\, d_z]^T$ vektor vanjskih sila poremećaja.

Četiri propelera kvadkoptera, vrte se kutnom brzinom ω_i i proizvode silu, usmjerenu prema gore,

$$f_i = k_i \omega_i^2. \quad \textbf{(2.9)}$$

Gdje je:

- i = 1,2,3,4,
- k_i = pozitivne konstante.

Matrica alokacija sila, povezuje četiri sile aktuatora letjelice i upravljačke signale.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} F_z \\ \tau_\Phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -l & 0 & l & 0 \\ 0 & l & 0 & -l \\ c & -c & c & -c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix}. \quad \textbf{(2.10)}$$

Gdje je:

- l udaljenost motora od centra mase
- c konstanta koja povezuje silu i moment

Tokom leta, kutevi phi i theta će biti blizu vrijednosti 0, pretpostavka malih kuteva, čime možemo matricu $\boldsymbol{\Omega}_B$, smatrati jediničnom tako da derivaciju Eulerovih kuteva $\dot{\boldsymbol{\Theta}}$ možemo aproksimirati vektorom kutnih brzina u okviru vezanom za tijelo **ω,** u kojem i mjerimo kutne brzine. Uz ove pretpostavke dinamički model kvadkopter a (2.1)-(2.6) možemo zapisati u sljedećem obliku koji je prikladniji za sintezu regulatora.

$$\begin{aligned} m\ddot{x} &= (c\Psi\, s\Theta\, c\Phi + s\Psi\, s\Phi)\, F_z + d_x, \\ m\ddot{y} &= (s\Psi\, s\Theta\, c - c\Psi\, s\Phi)\, F_z + d_y, \\ m\ddot{z} &= -mg + c\Phi\, c\Theta\, F_z + d_z. \end{aligned} \quad \textbf{(2.11)}$$

$$\begin{aligned} I_x\ddot{\Phi} &= \left(I_y - I_z\right)\dot{\Theta}\Psi + M_x + d_\Phi, \\ I_y\ddot{\Theta} &= (I_z - I_x)\Phi\Psi + M_y + d_\theta, \\ I_z\ddot{\Psi} &= \left(I_x - I_y\right)\dot{\Phi}\dot{\Theta} + M_z + d_\psi. \end{aligned} \quad \textbf{(2.12)}$$

Gdje su:

- $d_x, d_y, d_z, d_\Phi, d_\theta, d_\psi$ vanjski poremečaji.

3. Upravljački algoritam

Cilj nam je dobiti algoritam koji možemo koristiti za dovođenje letjelice u bilo koji položaj koji možemo, pošto imamo četiri aktuatora ne možemo ostvariti sva gibanja. Da bi letjelica išla u nekom od horizontalnih smjerova mora se nagnuti odnosno zakrenuti oko svoje osi.

Sinteza je postupak određivanja parametara sustava iz poznate pobude i odziva, dakle biramo pojačanja kojima ćemo dobiti željene performanse, tako da upravljačkim zakonima modificiramo osnovnu dinamiku sustava. Želja nam je upravljati letjelicom iz inercijalnog referentnog okvira, sa Zemlje.

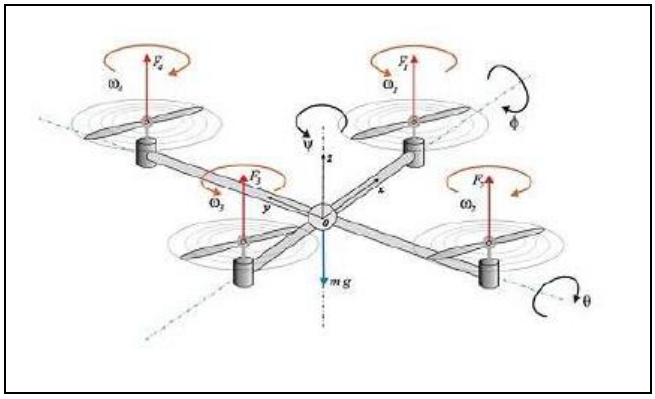
Analiza je postupak određivanja odziva sustava iz poznatih parametara i pobude. Poznavanjem dinamičkog modela možemo analizom vidjeti kako sustav reagira na pobudu uz poznavanje početnih uvjeta. To nam je korisno zato što možemo vidjeti kakve je naravi naš sustav.

Sinteza bi se mogla provoditi na punom dinamičkom modelu, ali je to vrlo kompleksno i iznad razine ovog rada. Stoga se u dinamički model uvode pojednostavljenja kojima bi se dinamički model pojednostavio, a s time i sinteza.

U radu je korišteno 3 modela, na najjednostavnijem je vršena sinteza regulatora, a na drugim modelima, kompleksnijim, su ispitani upravljački algoritmi bazirani na pojednostavljenom modelu. Slijedi opis pojednostavljenja i svih modela.

3.1. Pojednostavljeni dinamički modeli

U okviru ovog završnog rada, zanemaruju se vanjski poremećaji.



Slika 3 Model drona [5]

$$
\begin{aligned}
m\ddot{x} &= (c\Psi\ s\Theta\ c\Phi + s\Psi\ s\Phi)\ F_z \\
m\ddot{y} &= (s\Psi\ s\Theta\ c - c\Psi\ s\Phi)\ F_z \\
m\ddot{z} &= -mg + c\Phi\ c\Theta\ F_z \\
I_x\ddot{\Phi} &= \left(I_y - I_z\right)\dot{\Theta}\dot{\Psi} + M_x \\
I_y\ddot{\Theta} &= (I_z - I_x)\Phi\dot{\Psi} + M_y \\
I_z\ddot{\Psi} &= \left(I_x - I_y\right)\dot{\Phi}\dot{\Theta} + M_z
\end{aligned}
\qquad \textbf{(3.1)}
$$

Gdje su:

- Φ, θ, ψ – tri Eulerova kuta
- x,y,z – pozicije centra mase u inercijalnom referentnom okviru
- F_z – okomita sila u referentnom okviru vezanom uz tijelo
- M_x, M_y, M_z – momenti oko glavnih osi krutog tijela
- c - cos()
- s – sin()

U simulaciji ovo će biti model 2, ista pretpostavka o malim kutevima znači da će matrica sa slike 5 postati približno jednaka jediničnoj dijagonalnoj matrici stoga ćemo moći mjerene vrijednosti s žiroskopa direktno koristiti kao kutne brzine Eulerovih kuteva:

$$
\begin{bmatrix} \dot{\Phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} p \\ q \\ r \end{bmatrix} \qquad \textbf{(3.2)}
$$

Nadalje uvođenjem pretpostavke malih kuteva u translacijske jednadžbe, što znači da je umnožak dvaju malih kuteva ili njihovih derivacija približno jednak 0, dobijemo:

$$
\begin{aligned}
m\ddot{x} &= \theta\ F_z \\
m\ddot{y} &= -\Phi\ F_z \\
m\ddot{z} &= -mg + F_z \\
I_x\ddot{\Phi} &= M_x \\
I_y\ddot{\Theta} &= M_y \\
I_z\ddot{\Psi} &= M_z
\end{aligned}
\qquad \textbf{(3.3)}
$$

Ovo još uvijek nije linearni model, zbog umnoška upravljačke variable i variable stanja, takav model se naziva bilinearni model.

U daljem tekstu će F_z biti zamijenjen s U_1, M_x s U_2, My s U_3 i M_z s U_4.

Da bismo bilinearni model potpuno linearizirali uvodimo pojednostavljenje da je $U_1=mg+\hat{U}_1$ (feedforward) gdje je $\hat{U}=0$ za pretpostavku da letjelica radi u stanju bliskom letjenju. Dobijemo potpuno linearizirani model:

$$\begin{aligned} \ddot{x} &= g\theta \\ \ddot{y} &= -g\,\Phi \\ \ddot{z} &= \frac{1}{m}\,\hat{U}_1 \\ I_x\ddot{\Phi} &= U_2 \\ I_y\ddot{\Theta} &= U_3 \\ I_z\ddot{\Psi} &= U_4 \end{aligned} \qquad \textbf{(3.4)}$$

Model 3 u simulacijama, model na kojem provodimo sintezu regulatora u nastavku.

3.2. Sinteza linearnog regulatora

Kako želimo upravljati letjelicom sa Zemlje, želimo upravljati translacijom po x,y i z i rotacijom oko z, ψ ili ako se upravlja ručno intuitivnije je upravljanje brzinama.

Pogodno je podjeliti sustav u četiri podsustava i riješiti svaki zasebno.

- Podsustav 1:

$$\ddot{z} = \frac{1}{m}\,\hat{U}_1 \qquad \textbf{(3.5)}$$

- Podsustav 2:

$$\ddot{\Psi} = \frac{1}{I_z}U_4 \qquad \textbf{(3.6)}$$

- Podsustav 3:

$$\begin{aligned} \ddot{x} &= g\theta \\ \ddot{\Theta} &= \frac{1}{I_y}U_3 \\ x^{(4)} &= \frac{g}{I_y}U_3 \end{aligned} \qquad \textbf{(3.7)}$$

- Podsustav 4:

$$\begin{aligned} \ddot{y} &= -g\,\Phi \\ \ddot{\Phi} &= \frac{1}{I_x}U_2 \\ y^{(4)} &= -\frac{g}{I_x}U_2 \end{aligned} \qquad \textbf{(3.8)}$$

3.3. Regulacija po poziciji

Ako imamo sustav želimo ga koristit za neku namjenu, stoga ako imamo i dinamički model, možemo kontrolirati sustav da primjerice održava temperaturu u prostoriji konstantnom, u naftnu bušotinu želimo bušiti brzinom koja najbolje odgovara tipu stijene koja je na nekoj dubini ili želimo da naša bezpilotna letjelica leti nekom brzinom. Da bismo to postigli trebamo imati regulator, kojemu je cilj pogrešku (razliku željene i trenutačne veličine) svesti na nulu. Neophodno je imati negativnu povratnu vezu, što znači da mjerimo stanje koje reguliramo i to mjerenje vraćamo u regulator.

Kao što je prije u tekstu navedeno radi se o dronu koji bi bio namijenjen dostavljanju, što znači da ne znamo masu same letjelice. Ako bismo primijenili PD regulator, ne bismo dobili dobre odnosno željene rezultate, zbog toga što je PD regulator osjetljiv na netočnosti u modelu (nepoznavanje mase), nije dobar za točno slijeđenje promjenjivih ulaznih veličina i ne može kompenzirati poremećaje, stoga moramo odabrati PID regulator kojim ćemo riješiti trajno regulacijsko odstupanje. No u ovom radu ćemo pretpostaviti poznavanje mase i koristiti PD regulator, s kompenzacijom gravitacije.

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)dt + K_D \frac{de(t)}{dt} \tag{3.9}$$

Gdje je:

- *e(t)* - regulacijska pogreška, odnosno razlika između trenutne vrijednosti neke variable i njene tražene vrijednosti
- K_p – pojačanje proporcionalnog člana, P djelovanje generira vrijednost proporcionalnu vrijednosti pogreške
- K_D –pojačanje derivativnog člana, D djelovanje generira vrijednost na temelju brzine promjene pogreške, omogućuje brzo reagiranje sustava ako je pogreška mala, a počinje brzo rasti
- K_I – pojačanje integralnog člana, I djelovanje predstavlja sumiranje greške u vremenu, tako da se vrijednost povećava dok greška ne dođe do nule.

PID regulator je najčešći regulator u praksi, zbog svoje jednostavnosti i efikasnosti. Pravilnim izborom pojačanja može se postići većina regulacijskih zahtjeva u industriji. U ovom radu pojačanje će se birati metodom odabira polova.

Većina sustava je nelinearna i linearni regulatori se mogu koristit ukoliko se dinamički model linearizira oko neke radne točke, ukoliko to nije moguće potrebno je primijeniti neke naprednije nelinearne regulatore.

- Podsustav 1

  Jednadžba 3.5, referentno stanje je z_d, a pogreška:

$$\tilde{z} = z - z_d. \tag{3.10}$$

  Kada raspišemo 3.5 s 3.10, odnosno derivacijama iste:

$$\ddot{\tilde{z}} + \ddot{z}_d = \frac{1}{m}\widehat{U_1}. \tag{3.11}$$

  Cilj regulacije je globalna asimtotska stabilnost, što znači da regulacijska pogreška teži u 0, željena dinamika pogreške:

$$a_2\ddot{\tilde{z}} + a_1\dot{\tilde{z}} + a_0\tilde{z} = 0. \tag{3.12}$$

  Odabirom odgovarajuće upravljačke variable $\hat{U}_1$, PD regulator:

$$\hat{U}_1 = m\left(\ddot{z}_d - K_P\tilde{z} - K_D\dot{\tilde{z}}\right), \tag{3.13}$$

$$\ddot{\tilde{z}} + K_D\dot{\tilde{z}} + K_P\tilde{z} = 0. \tag{3.14}$$

  3.14 Laplaceovom transformacijom prebacimo u kompleksnu domenu:

$$s^2 + K_D s + K_P = 0 = (s - s_1)(s - s_2). \tag{3.15}$$

  Odabirom polova s_1, s_2 određujemo pojačanja. Kako bismo dobili aperiodski odziv biramo realne negativne polove, a pojačanje dobijmo na sljedeći način.

$$\begin{aligned} K_D &= -(s_1+s_2), \\ K_P &= s_1 s_2. \end{aligned} \tag{3.16}$$

- Podsustav 2

  Jednadžbe 3.6

$$\tilde{\psi} = \psi - \psi_d, \tag{3.17}$$

Ψ_d- referentni kut oko z osi

kao i za prethodni, odabiremo upravljački algoritam:

$$U_4 = I_z\left(-K_p\tilde{\psi} - K_D\dot{\tilde{\psi}} + \ddot{\psi}_d\right). \tag{3.18}$$

Kada 3.16 ubacimo u 3.6 i prebacimo sve na jednu stranu dobijemo dinamiku greške:

$$\ddot{\tilde{\psi}} + K_D\dot{\tilde{\psi}} + K_P\tilde{\psi} = 0 = (s - s_1)(s - s_2), \tag{3.19}$$

3.19 u kompleksnoj domeni

$$s^2 + K_D s + K_P = 0 = (s - s_1)(s - s_2). \tag{3.20}$$

Polove biramo na isti način kao i u prethodnome slučaju.

$$\begin{aligned} K_P &= s_1 s_2, \\ K_D &= -(s_1 + s_2). \end{aligned} \tag{3.21}$$

- Podsustav 3 i 4

  U suštini identični zbog simetričnosti konstrukcije, jednadžbe 3.7 i 3.8.

$$\tilde{x} = x - x_d, \tag{3.22}$$

$$U_3 = \frac{I_y}{g}\left(x_d^{(4)} - K_3\dddot{\tilde{x}} - K_2\ddot{\tilde{x}} - K_1\dot{\tilde{x}} - K_0\tilde{x}\right), \tag{3.23}$$

$$\breve{x}^{(4)} + K_3\dddot{\tilde{x}} + K_2\ddot{\tilde{x}} + K_1\dot{\tilde{x}} + K_0\tilde{x} = 0, \tag{3.24}$$

$$s^4 + K_3 s^3 \pm K_2 s^2 + K_1 s + K_0 = 0 = (s - s_1)(s - s_2)(s - s_3)(s - s_4). \tag{3.23}$$

Biramo polove na isti način kao i prije. Ista pojačanje se koriste i u podsustavu 4.

$$\tilde{y} = y - y_d, \tag{3.24}$$

$$U_2 = \frac{I_x}{g}\left(+y_d^{(4)} - K_3\dddot{\tilde{y}} - K_2\ddot{y} - K_1\dot{y} - K_0\tilde{y}\right), \tag{3.25}$$

$$y^{(4)} + K_3\dddot{\tilde{y}} + K_2\ddot{\tilde{y}} + K_1\dot{\tilde{y}} + K_0\tilde{y} = 0, \tag{3.26}$$

$$s^4 + K_3s^3 \pm K_2s^2 + K_1s + K_0 = 0 = (s - s_1)(s - s_2)(s - s_3)(s - s_4). \tag{3.27}$$

Sveukupni regulator po poziciji:

$$\begin{aligned} U_1 &= m\left(\ddot{z}_d + g - K_P\tilde{z} - K_D\dot{\tilde{z}}\right), \\ U_2 &= \frac{I_x}{g}\left(+y_d^{(4)} - K_3\dddot{\tilde{y}} - K_2\ddot{\tilde{y}} - K_1\dot{\tilde{y}} - K_0\tilde{y}\right), \\ U_3 &= \frac{I_y}{g}\left(+x_d^{(4)} - K_3\dddot{\tilde{x}} - K_2\ddot{\tilde{x}} - K_1\dot{\tilde{x}} - K_0\tilde{x}\right), \\ U_4 &= I_z\left(-K_p\tilde{\psi} - K_D\dot{\tilde{\psi}} + \ddot{\psi}_d\right). \end{aligned} \tag{3.28}$$

3.4. Regulacija po brzini

Znatno jednostavniji slučaj, pogotovo kad je pretpostavka poznavanje mase letjelice, ovaj regulator, odnosno dobivene upravljačke algoritme ćemo koristiti u praktičnome djelu ovog rada. Pokušat ću napraviti drona kojeg bi svatko mogao voziti preko daljinskog, dajući jednostavne, intuitivne komande. Zadane komande su brzine u smjerovima koordinatnih osi, konstantne vrijednosti tako da je derivacija reference jednaka nuli.

$$\dot{x}_d = 0 \tag{3.29}$$

Tako i za ostale. Postupak je isti kao i u prethodnom poglavlju, ali razinu jednostavniji. Dok smo prije imali akceleraciju kao drugu derivaciju pomaka, sade dinamiku sustava možemo zapisati na ovaj način. Za svaki podsustav ponavljamo postupak kao i prije, dobivamo sustave

za jedan nižeg reda, unosom pretpostavke o poznavanju mase, regulacija po z osi se može izvesti običnim D regulatorom s kompenzacijom gravitacijskog djelovanja.

- Podsustav 1

$$\dot{v}_z = \frac{1}{m}\,\hat{\mathrm{U}}_1 \tag{3.30}$$

Odabiremo upravljačku varijablu:

$$\hat{\mathrm{U}}_1 = m(-K_P\,\tilde{v}_z) \tag{3.31}$$

I riješimo upravljanje u smjeru z osi, jedi uvjet je da je K_P veći od 0.

- Podsustav 2:

$$\dot{\omega}_z = \frac{1}{I_z} U_4 \tag{3.32}$$

Odabiremo upravljačku varijablu:

$$U_4 = -I_z\,(K_P\,\tilde{\omega}_z) \tag{3.33}$$

i riješimo upravljanje oko z osi, jedi uvjet je da je K_P veći od 0.

- Podsustav 3:

$$\begin{aligned} \dot{v}_x &= g\theta \\ \ddot{\Theta} &= \frac{1}{I_y} U_3 \\ \dddot{v}_z &= \frac{g}{I_y} U_3 \end{aligned} \tag{3.34}$$

- Podsustav 4:

$$\begin{aligned} \dot{v}_y &= -g\,\Phi \\ \ddot{\Phi} &= \frac{1}{I_x} U_2 \\ \dddot{v}_y &= -\frac{g}{I_x} U_2 \end{aligned} \tag{3.35}$$

Podsustavi 3 i 4 su opet u principu jednaki, biramo upravljačku varijablu:

$$U_2 = -\frac{I_x}{g}\left(-K_2\ddot{\tilde{v}}_y - K_1\dot{\tilde{v}}_y - K_0\tilde{v}_y\right) \quad \textbf{(3.36)}$$

Uvrštavanje 3.29 u 3.28, dobijmo dinamiku pogreške:

$$\dddot{\tilde{v}}_y + K_2\ddot{\tilde{v}}_y + K_1\dot{\tilde{v}}_y + K_0\tilde{v}_y = 0 \quad \textbf{(3.37)}$$

$$s^3 + K_2s^2 + K_1s + K_0 = 0 = (s - s_1)(s - s_2)(s - s_3) \quad \textbf{(3.38)}$$

Pravilnim odabirom polova možemo dobiti željeni odziv. Biramo negativne realne polove kako bismo dobili aperiodski odzib bez prebačaja.

$$K_2 = -(s_1 + s_2 + s_3), K_1 = s_1s_2 + s_1s_3 + s_2s_3, K_0 = -s_1s_2s_3 \quad \textbf{(3.39)}$$

Sve ukupni regulator po brzini:

$$\begin{aligned} U_1 &= m\,(g - K_P\,\tilde{v}_z) \\ U_2 &= -\frac{I_x}{g}\left(-K_2\ddot{\tilde{v}}_y - K_1\dot{\tilde{v}}_y - K_0\tilde{v}_y\right) \\ U_3 &= \frac{I_y}{g}\left(-K_2\ddot{\tilde{v}}_x - K_1\dot{\tilde{v}}_x - K_0\tilde{v}_x\right) \\ U_4 &= -I_z\,(K_P\,\tilde{\omega}_z) \end{aligned} \quad \textbf{(3.40)}$$

U sljedećem poglavlju će se pokazati odabrani polovi odnosno pojačanja. Kao i simulacije u Matlabu, numerički.

4. Simulacijski rezultati

U programskom paketu Matlab su implementirani dinamički modeli letjeli, puni model, pojednostavljeni i potpuno linearizirani model na kojem je rađena sinteza regulatora. Isto tako implementirani su i upravljački algoritmi. Numerička simulacija provedena je pomoću Runde-Kuta iterativne metode kojom se dobivaju približna rješenja skupa diferencijalnih jednadžbi prvog reda. Ukupno vrijeme simulacije je 40 sekundi.

Za simulaciju je potrebno poznavati fizičke parametre letjelice, masa i momenti inercije. Ti podaci su dobiveni iz programa za 3D modeliranje SolidWorks, u vrijeme ovog rada i provedbe ovih simulacija na dronu nije bilo gornjeg poklopca i skija za sljetanje.

Simulacija je provedena na svim dinamičkim modelima, za oba regulatora, po poziciji i po brzini. Regulatori su jednostavni i ne kompenziraju vanjske poremećaje kao što bi u stvarnosti bili udari vjetra, tako da nisu prikazani niti u simulacijama. Prvo će biti pokazani rezultati vezani uz regulator pozicije, a iza toga regulatora po brzini. Plavom crtom je prikazana trajektorija letjelice, a crvenom referentna trajektorija. Model 3 je potpuno linearizirani dinamički model, model 2 je pojednostavljeni dinamički model i model 3 je potpuni dinamički model. Početni uvjeti jednaki su 0 svi osi početnog stanja integratora kako letjelica ne bi propadala na početku.

4.1. Regulacija po poziciji

Zadana referentna trajektorija.

$$\begin{aligned} x_d &= \cos(0{,}5\ t) \\ y_d &= \sin(0{,}5\ t) \\ z_d &= 0{,}5t \\ \psi_d &= 0 \end{aligned} \qquad \textbf{(4.1)}$$

4.1.1. Prvi skup pojačanja

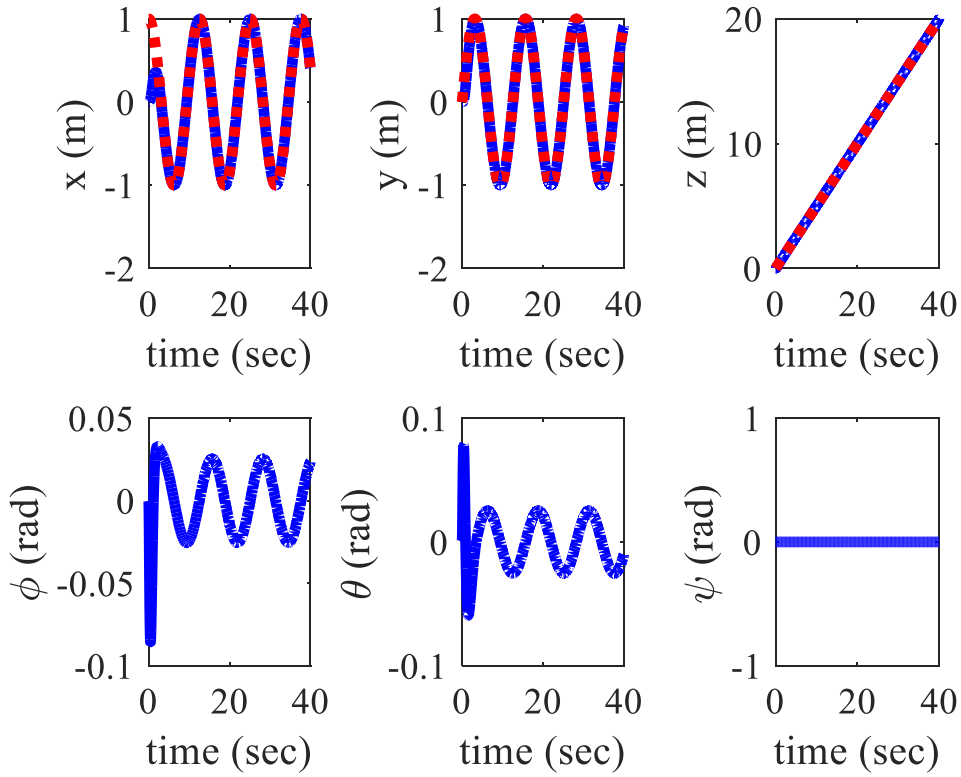
Odabrani polovi za regulator po osi Z i kuta ψ b_1= -2, b_2= -4, što daje:

- K_P = 8, K_D = 6.

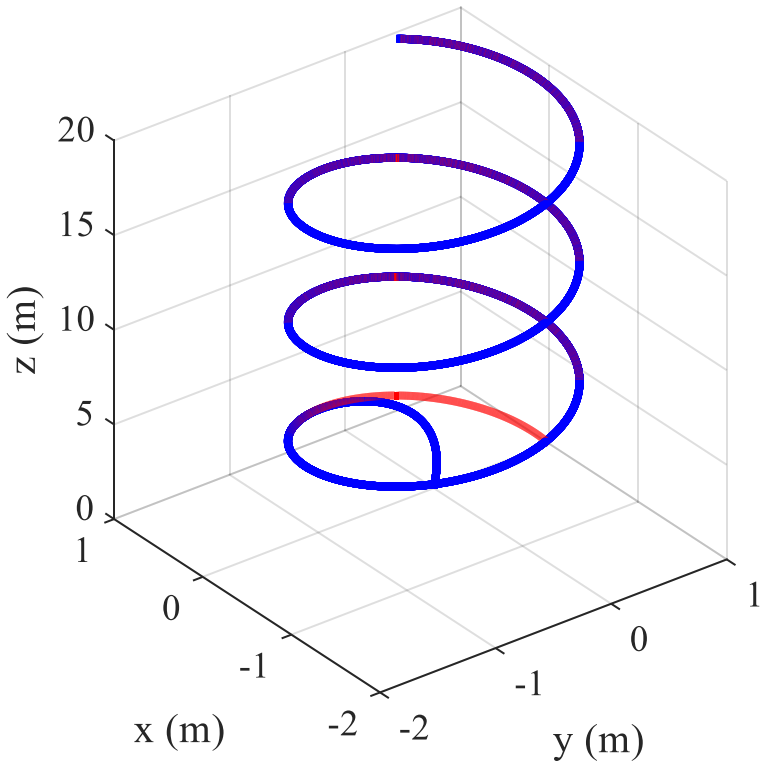
Odabrani polovi za regulator u smjeru osi x c_1= -2, c_2= -2, c_3 = -4, c_4= -5, što daje pojačanja: k_0=80, k_1=116, k_2=60, k_3=13. Ista pojačanja se koriste i za regulacuju Y osi.

4.1.1.1. Model 3

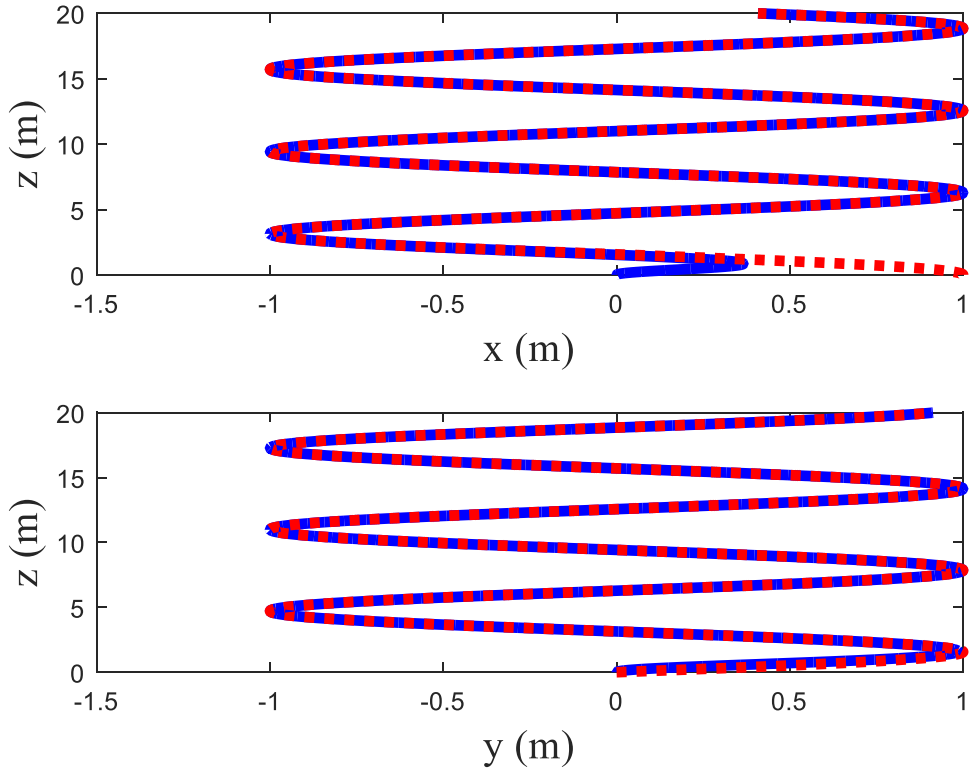
Slike 7-10 prikazuju rezultate simulacije, regulacije po poziciji s PD regulatorom s kompenzacijom gravitacije po z osi.



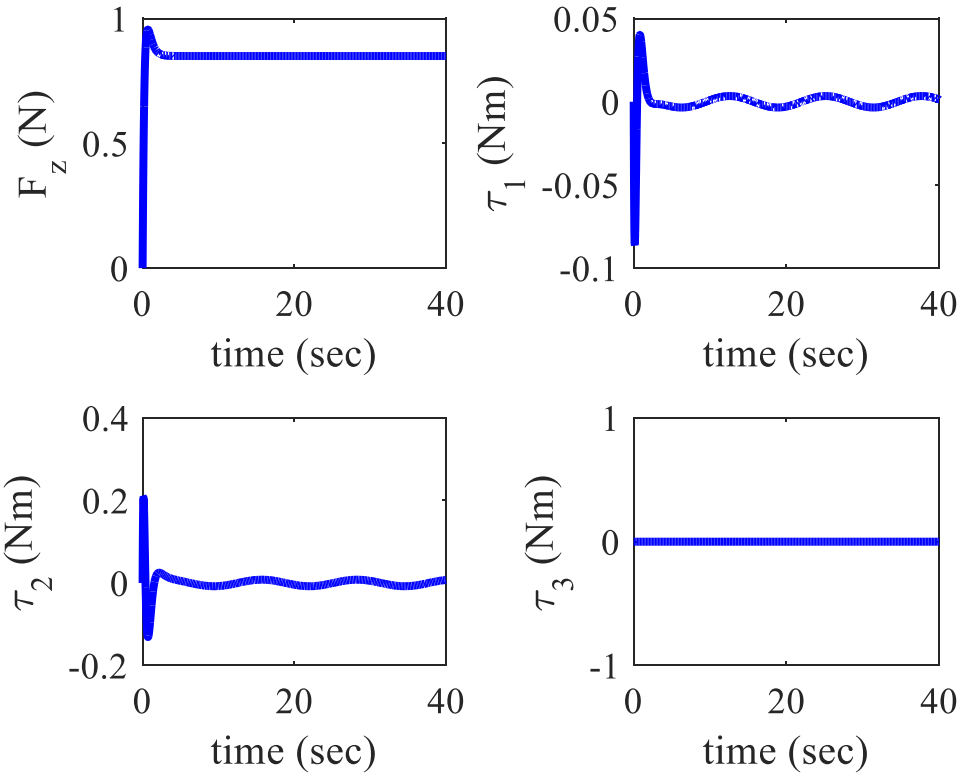
Slika 4 Odziv regulatora na zadane reference



Slika 5 3D praćenje trajektorije



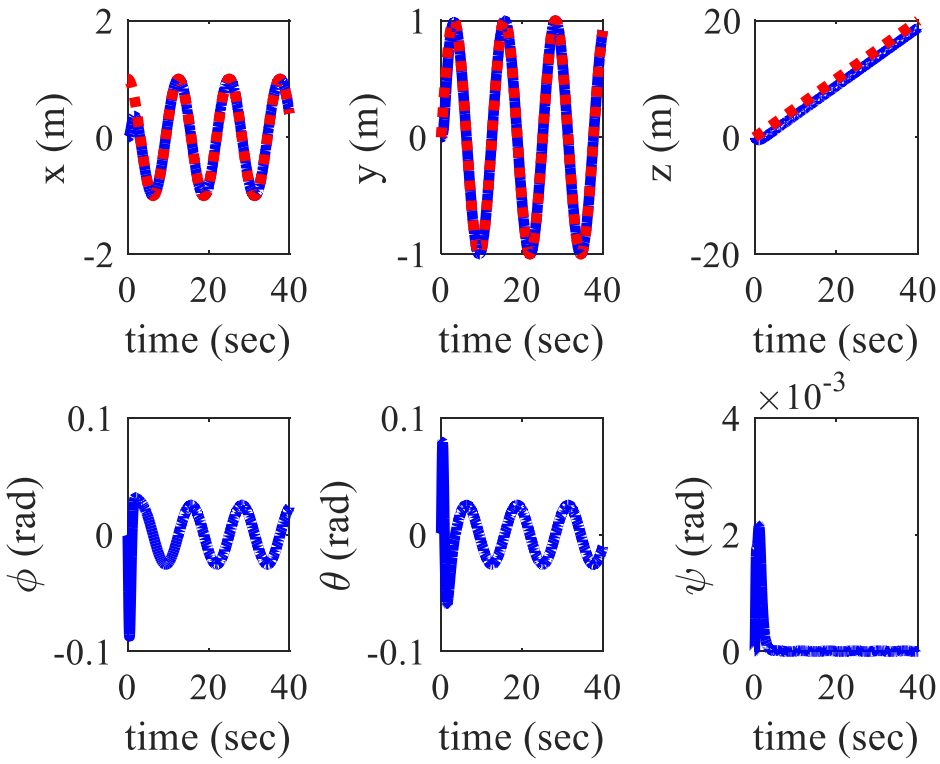
Slika 6 Ovisnost pozicija X i Y o Z



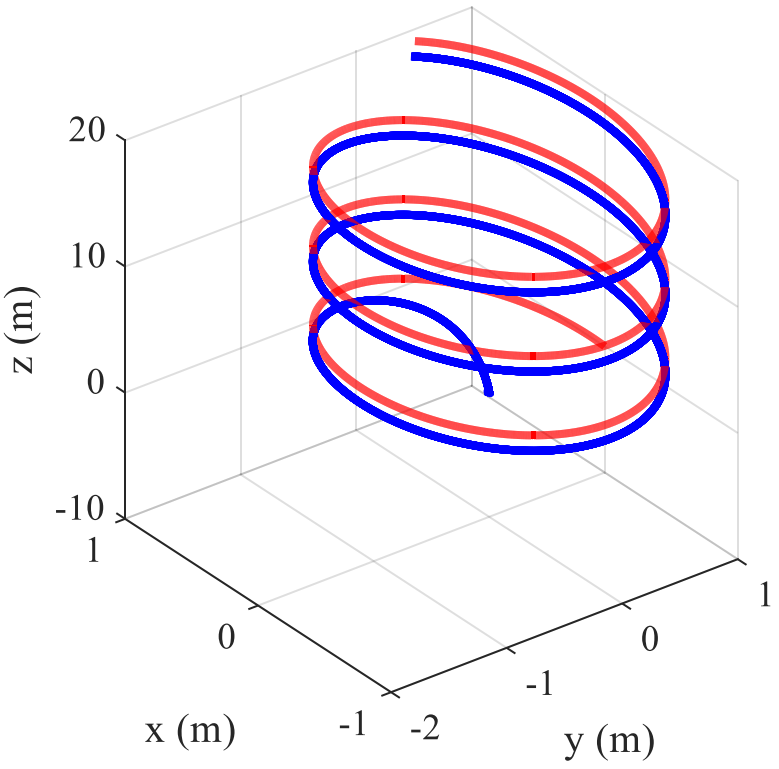
Slika 7 Upravljačke variable u vremenu

Kao i očekivano dobili smo dobre rezultate, linearni regulator, na potpuno lineariziranom modelu. Referentnu putanju dostiže u dvije sekunde, uz blage prebačaje.

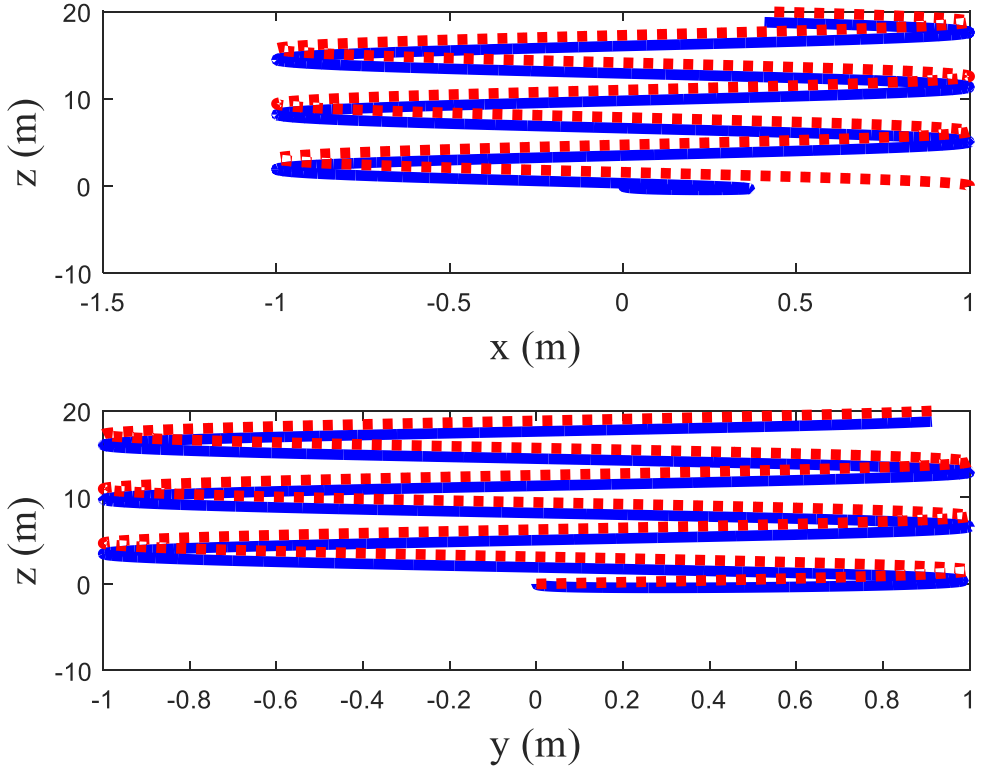
4.1.1.2. Model 2.



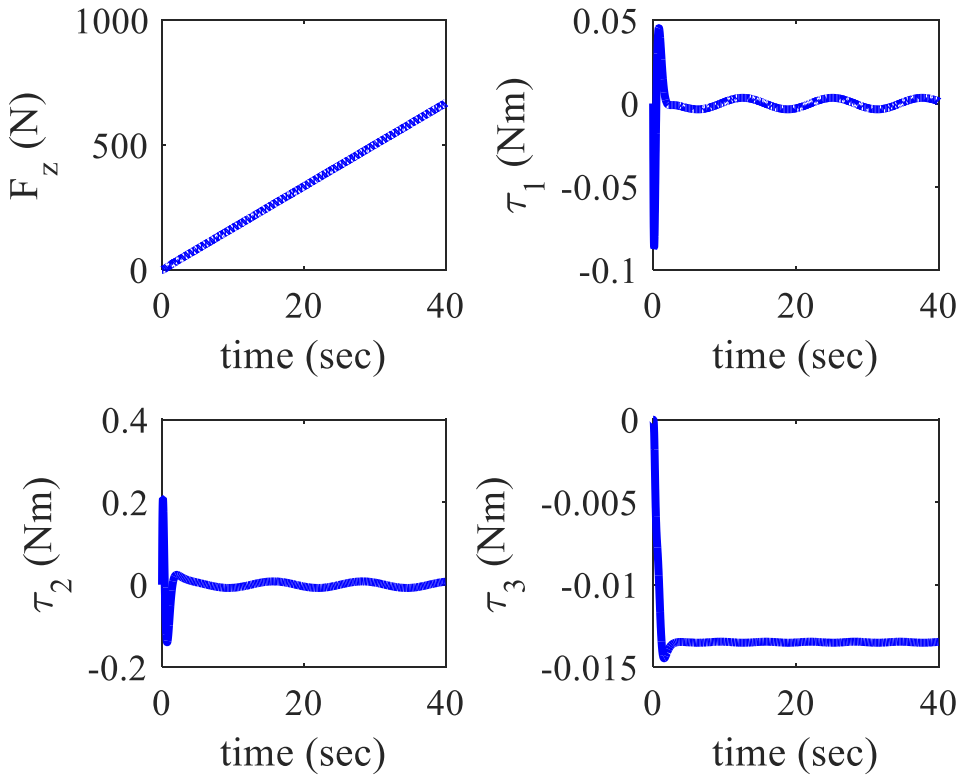
Slika 8 Odziv regulatora na zadane reference



Slika 9 3D praćenje trajektorije



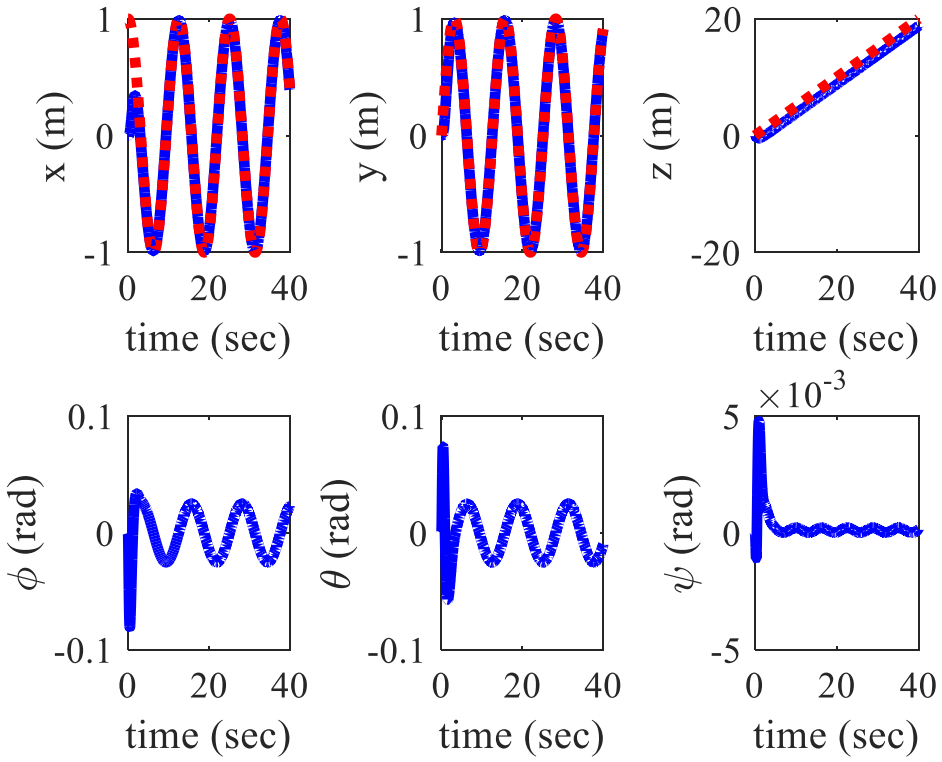
Slika 10 Ovisnost pozicija X i Y o Z



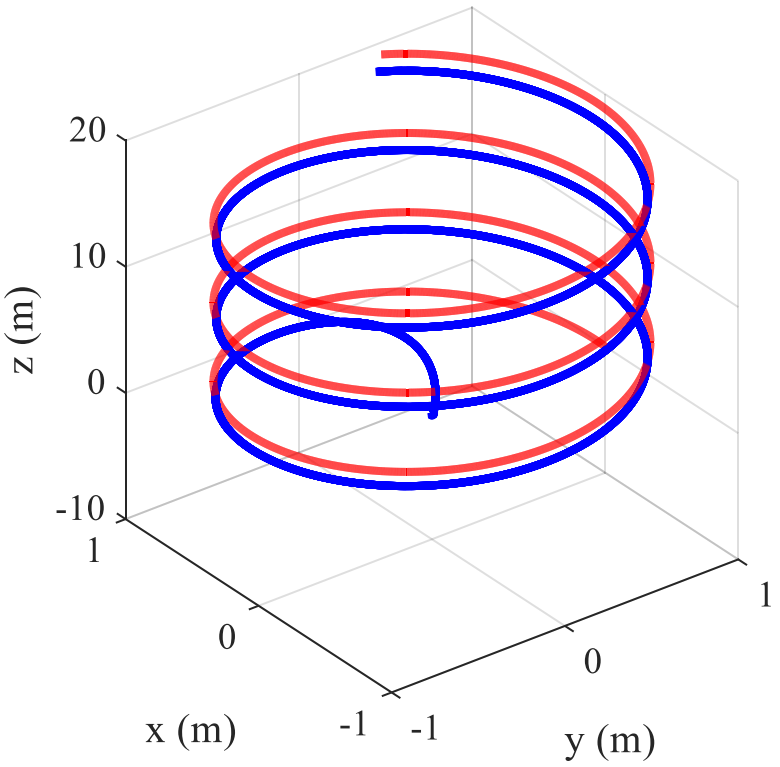
Slika 11 Upravljačke variable u vremenu

Na slikama 11- 14 prikazani su reziltati istog regulatora na modelu 2 koji je kompleksniji, vidi se odstupanje od reference.

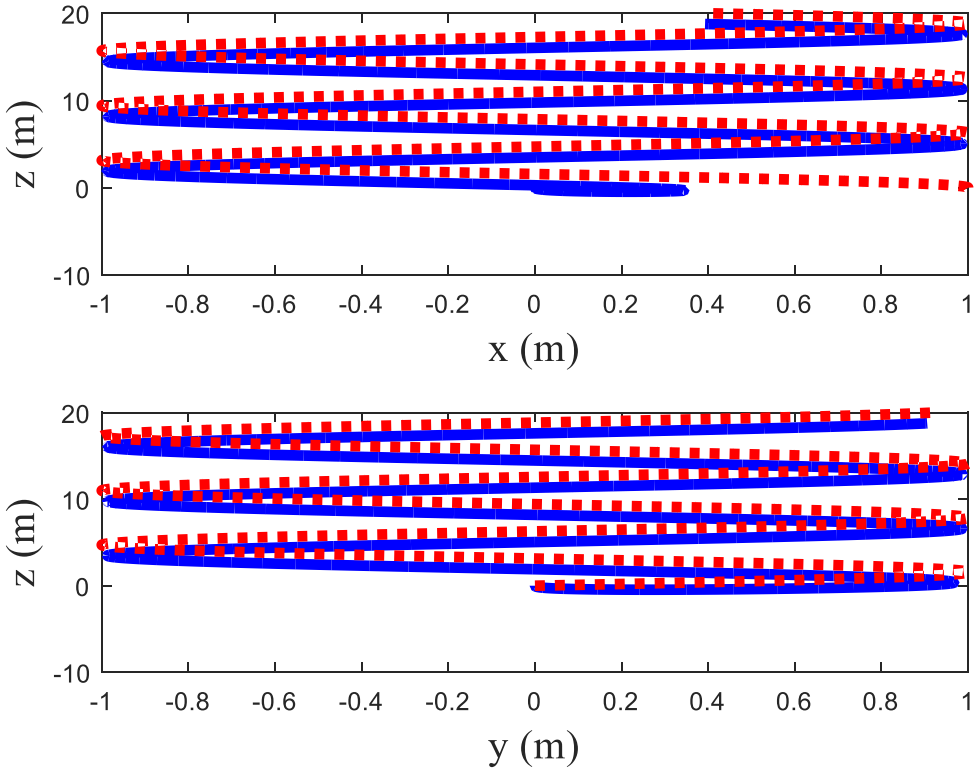
4.1.1.3. Model 1



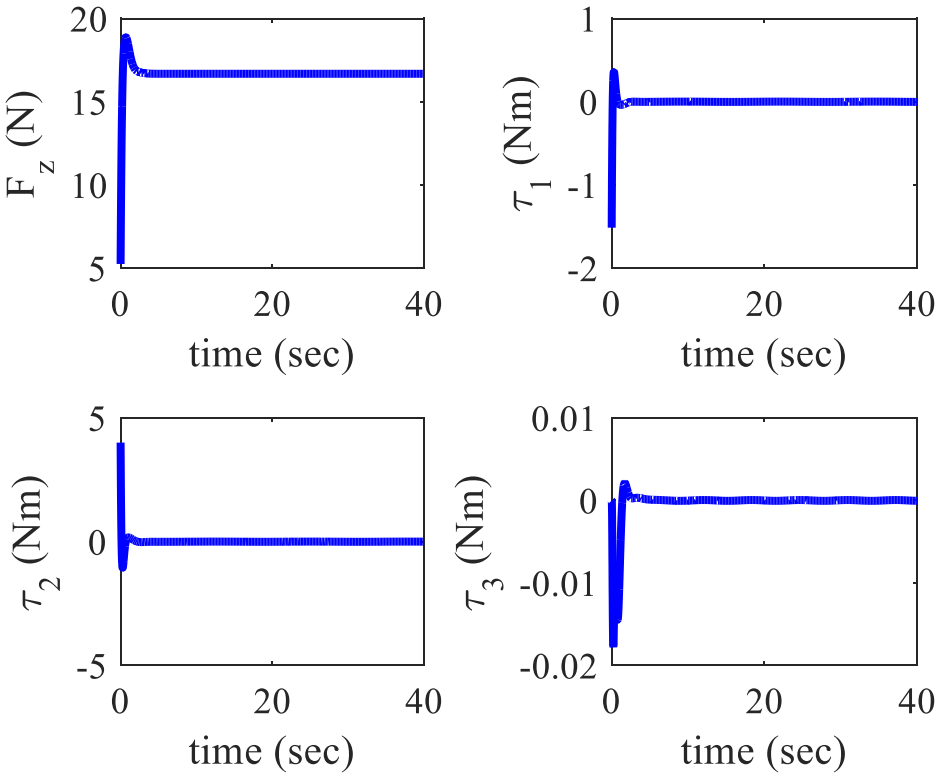
Slika 12 Odziv regulatora na zadane reference



Slika 13 3D praćenje trajektorije



Slika 14 Ovisnost pozicija X i Y o Z



Slika 15 Upravljačke variable u vremenu

Slike 15-18 prikazuju isti regulator na punom dinamičkom modelu. Zapažamo značajnu regulacijsku pogrešku. Regulator nije dovoljno dobar da kompenzira greške nastale linearizacijom modela.

4.1.2. Drugi skup pojačanja

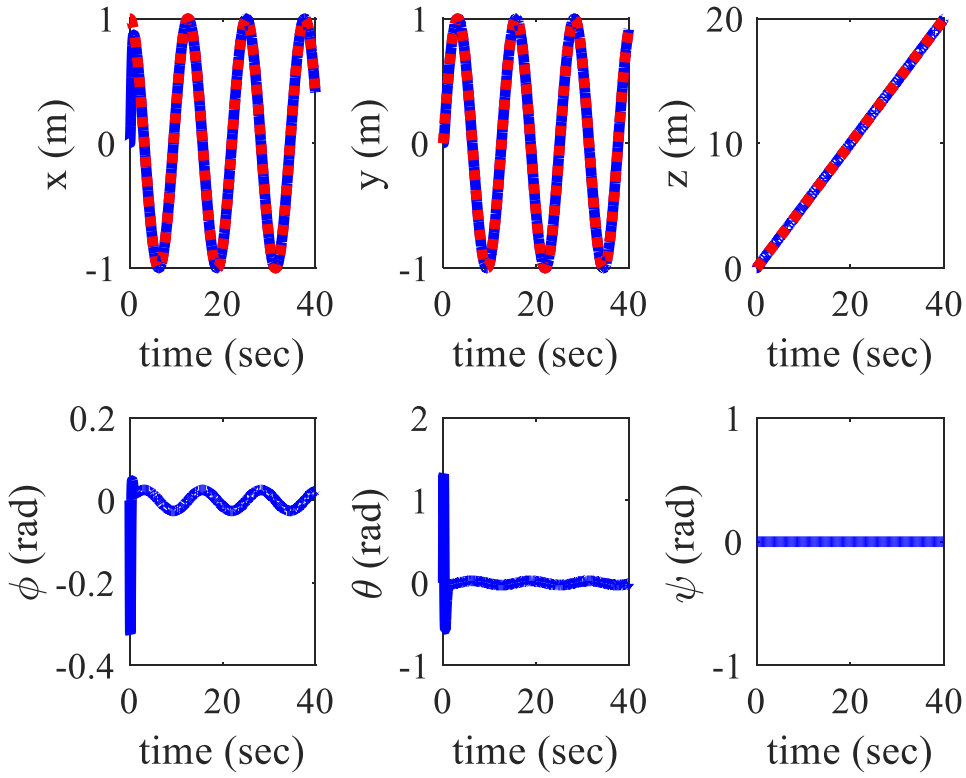
Odabrani polovi za regulator po osi Z i kuta ψ b_1= -12, b_2= -20, što daje:

- K_P = 240, K_D = 32.

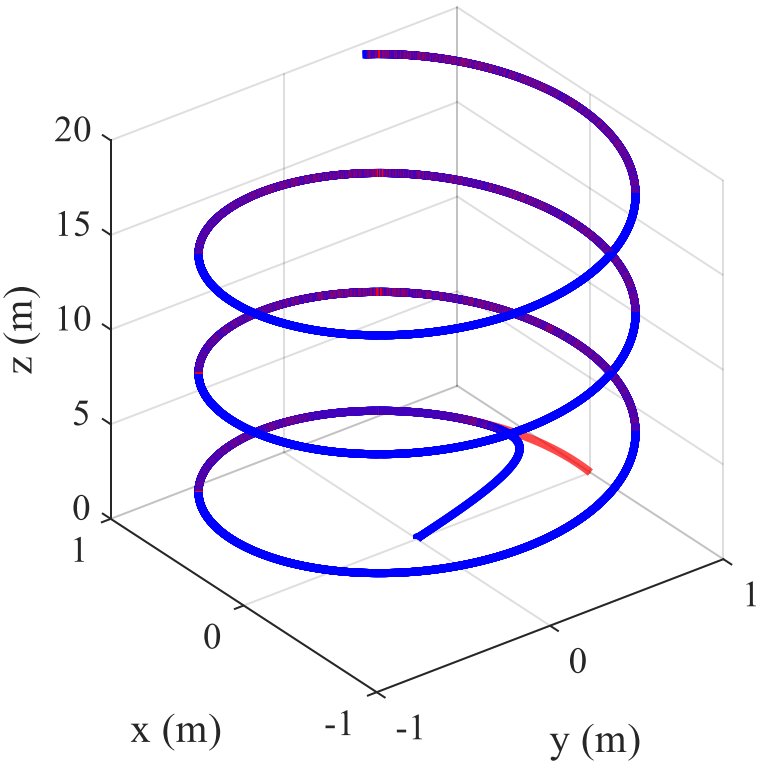
Odabrani polovi za regulator u smjeru osi x c_1= -7, c_2= -8, c_3 = -14, c_4= -15, što daje pojačanja: k_0=11760, k_1=4774, k_2=701, k_3=44. Ista pojačanja se koriste i za regulacuiju Y osi.

4.1.2.1. Model 3

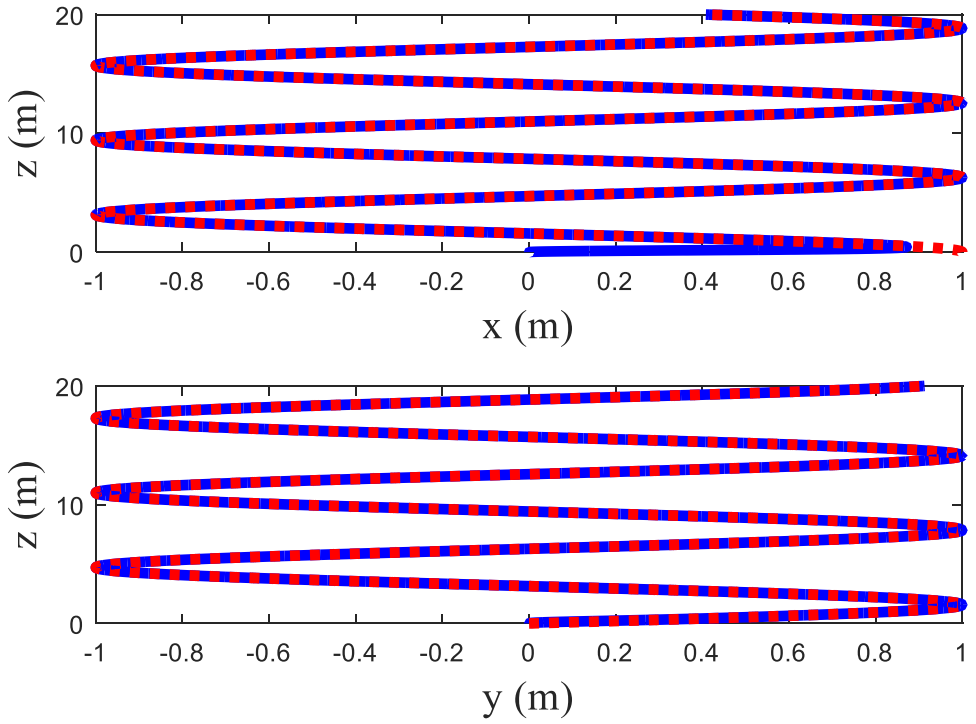
Slike 19 -22, prikazuju rezultate simulacije regulacije po poziciji za drugi set pojačanja, znatno većih. Očekuje se brži odziv i dobro praćenje reference.



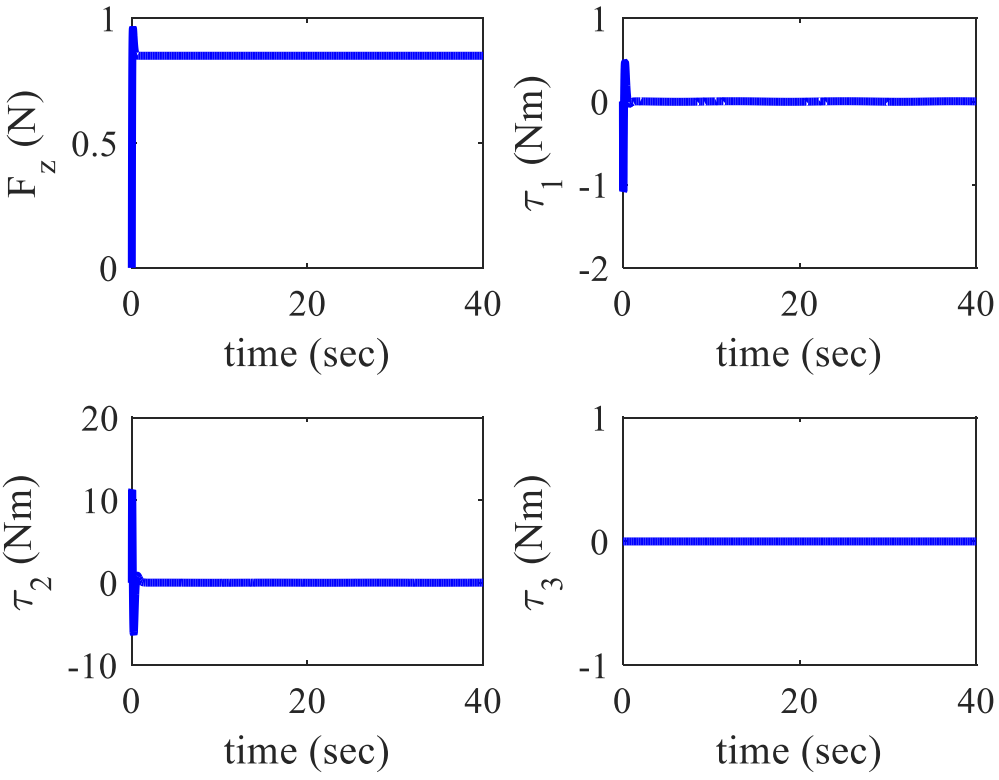
Slika 16 Odziv regulatora na zadane reference



Slika 17 3D praćenje trajektorije



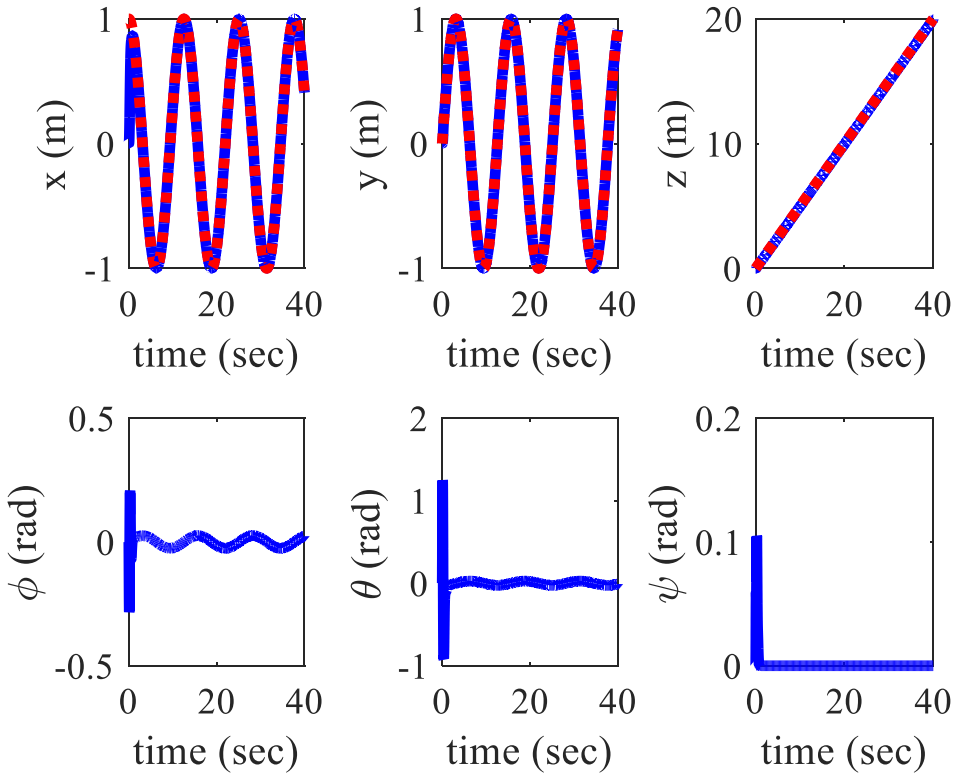
Slika 18 Ovisnost pozicija X i Y o Z



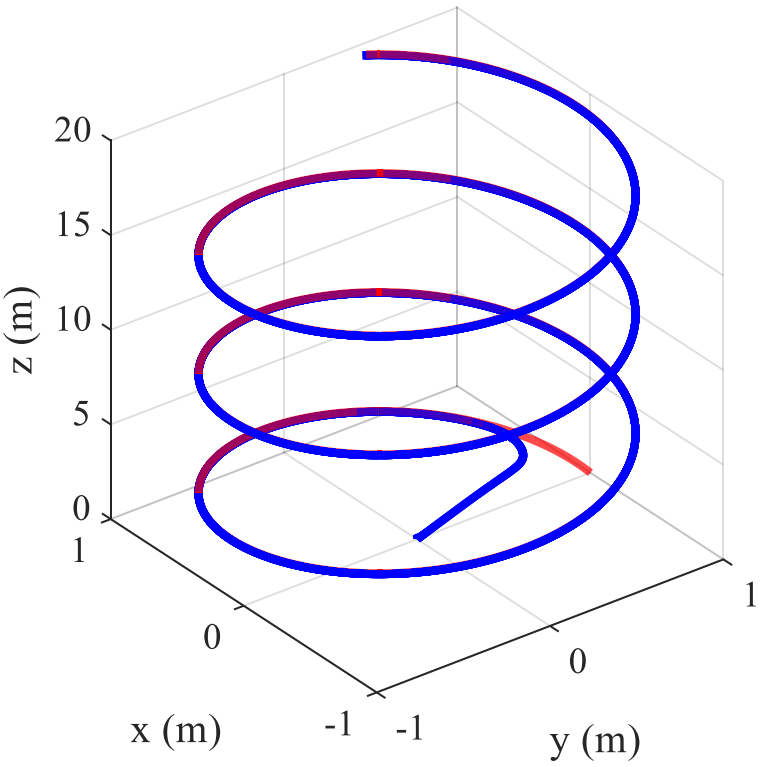
Slika 19 Upravljačke variable u vremenu

Zbog znatno većih pojačanja odziv je vidljivo brži, a i naprezanja upravljačkih varijabli su veća.

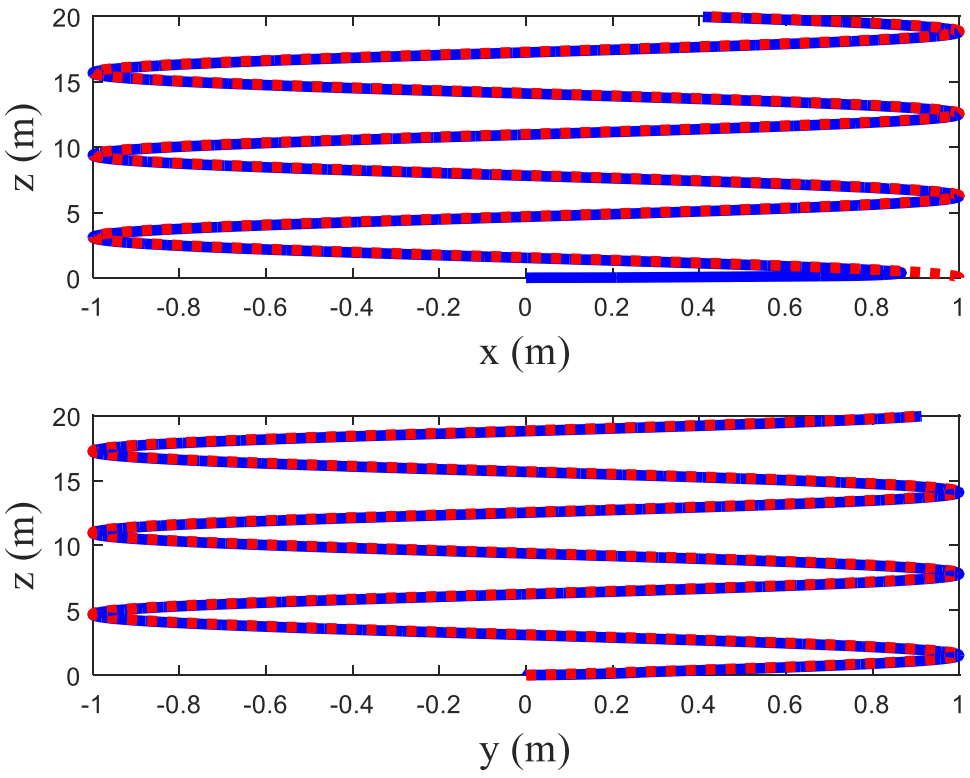
4.1.2.2. Model 2



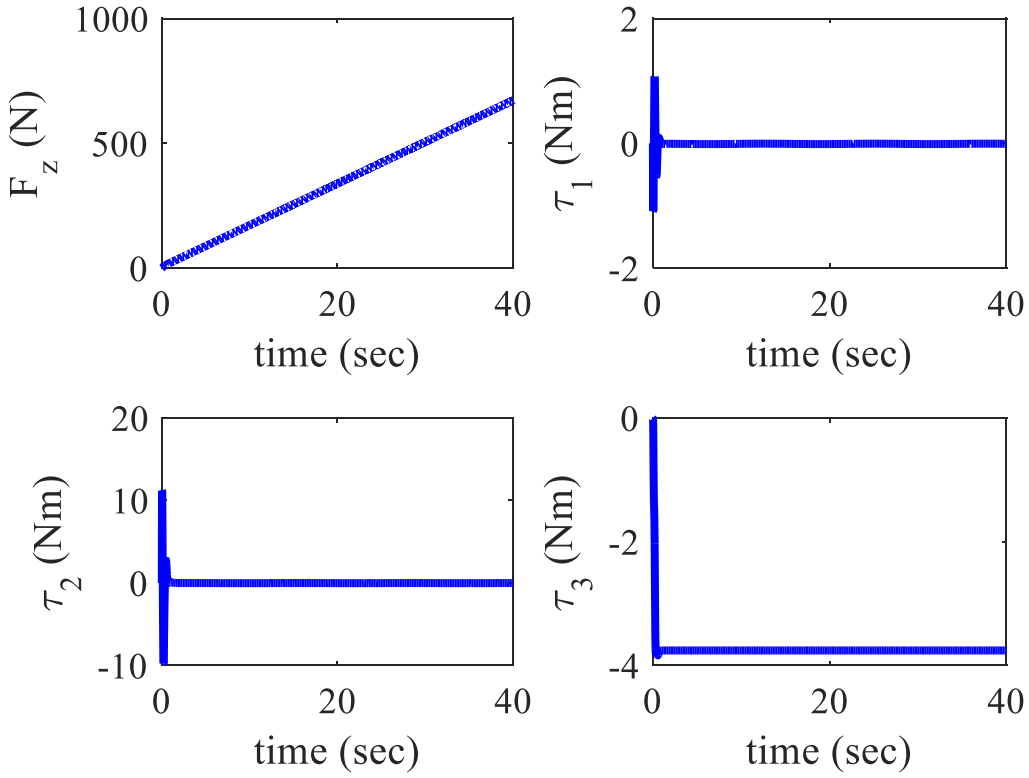
Slika 20 Odziv regulatora na zadane reference



Slika 21 3D praćenje trajektorije



Slika 22 Ovisnost pozicija X i Y o Z

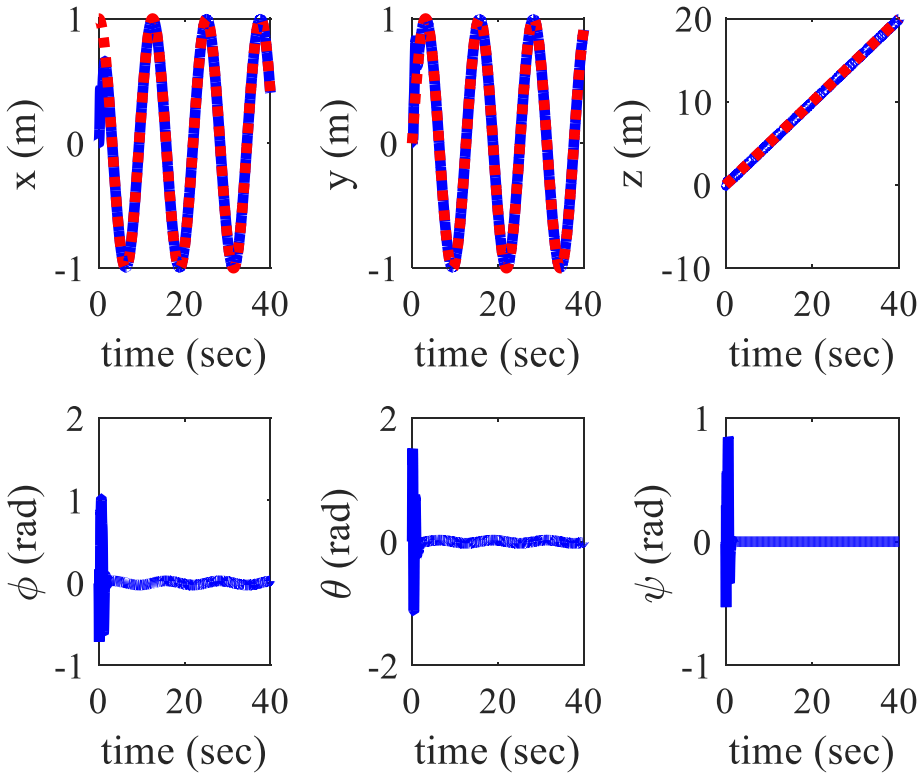


Slika 23 Upravljačke variable u vremenu

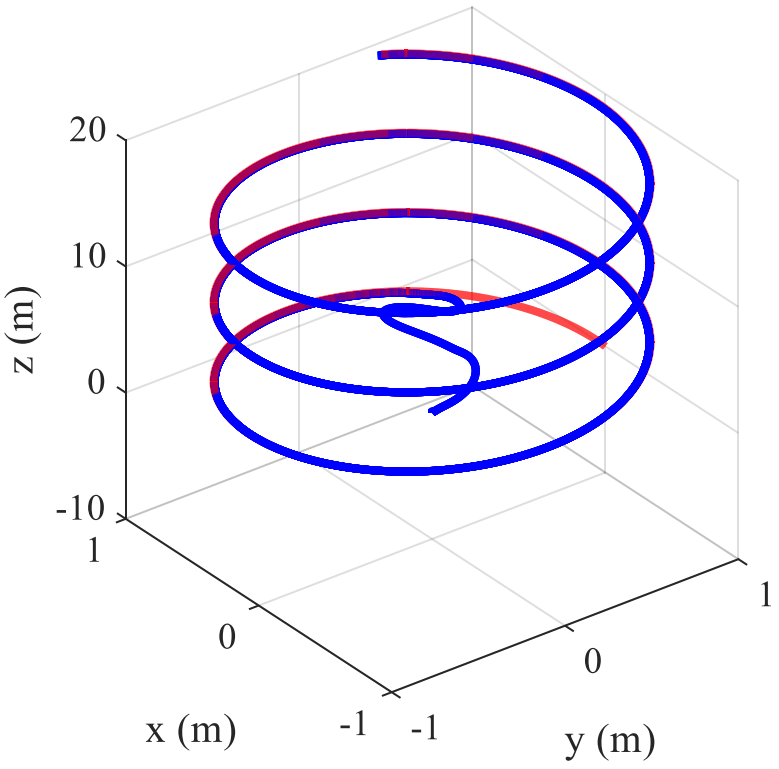
Slike 23 – 26 prikazuju rezultate na modelu 2, pojednostavljeni model, uz veća pojačanja regulatora. Veća pojačanja doprinijela su manjoj pogrešci.

4.1.2.3. Model 1.

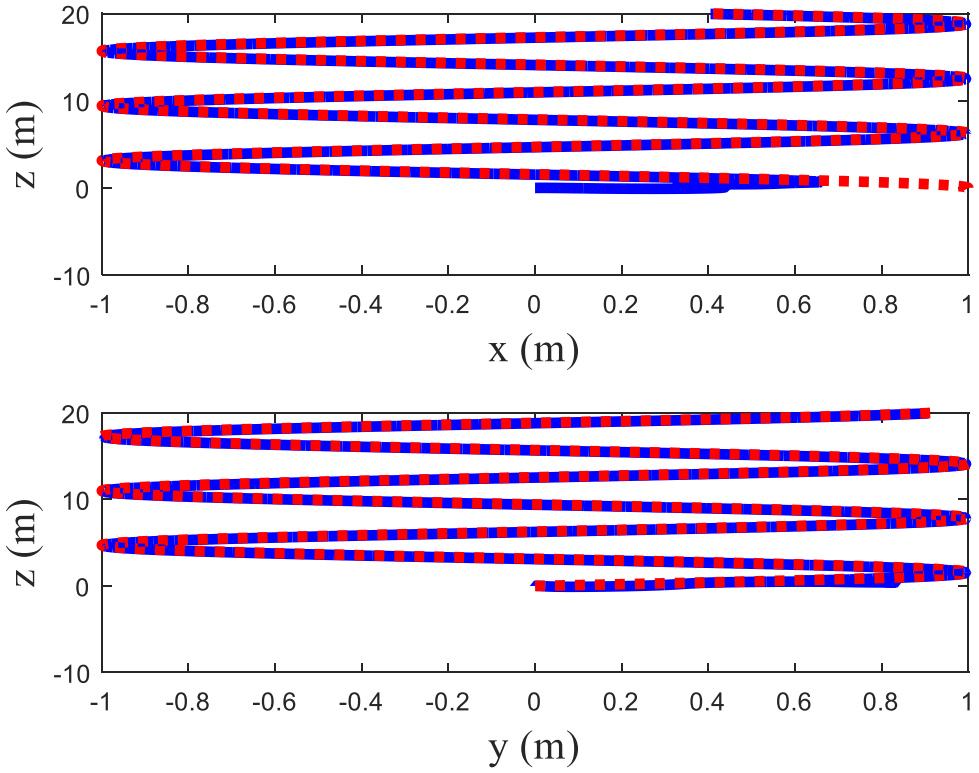
Slike 27- 30 prikazuju rezultate simulacija na potpunom dinamičkom modelu uz primjenu našeg linearnog regulatora.



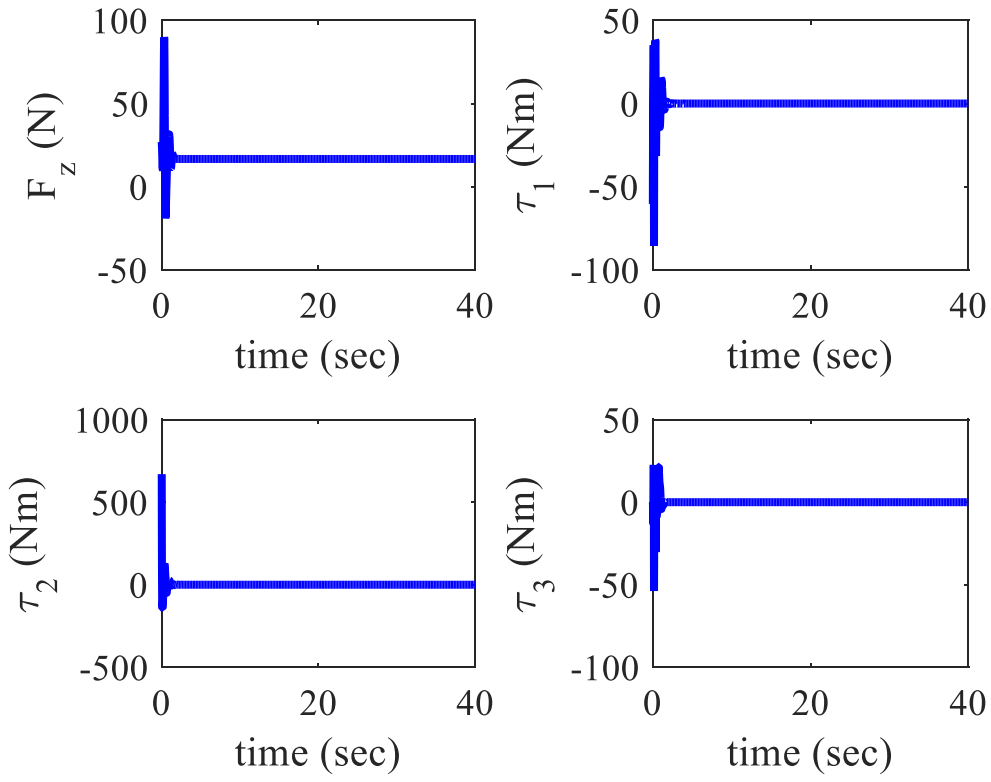
Slika 24 Odziv regulatora na zadane reference



Slika 25 3D praćenje trajektorije



Slika 26 Ovisnost pozicija X i Y o Z



Slika 27 Upravljačke variable u vremenu

4.2. Regulacija po brzini

Kako je i prije rečeno ovaj regulator je namijenjen za korištenje na dronu u sljedećem poglavlju. Želja je upravljati brzinama iz referentnog koordinatnog sustava, tako da će se u ovom djelu pokazati odziv regulatora na step pobudu amplitude 0.5 m/s.

4.2.1. Prvi set pojačanja

Ujedno i pojačanja koja će se koristiti i u našem dronu.

Pojačanje regulatora u smjeru osi Z je

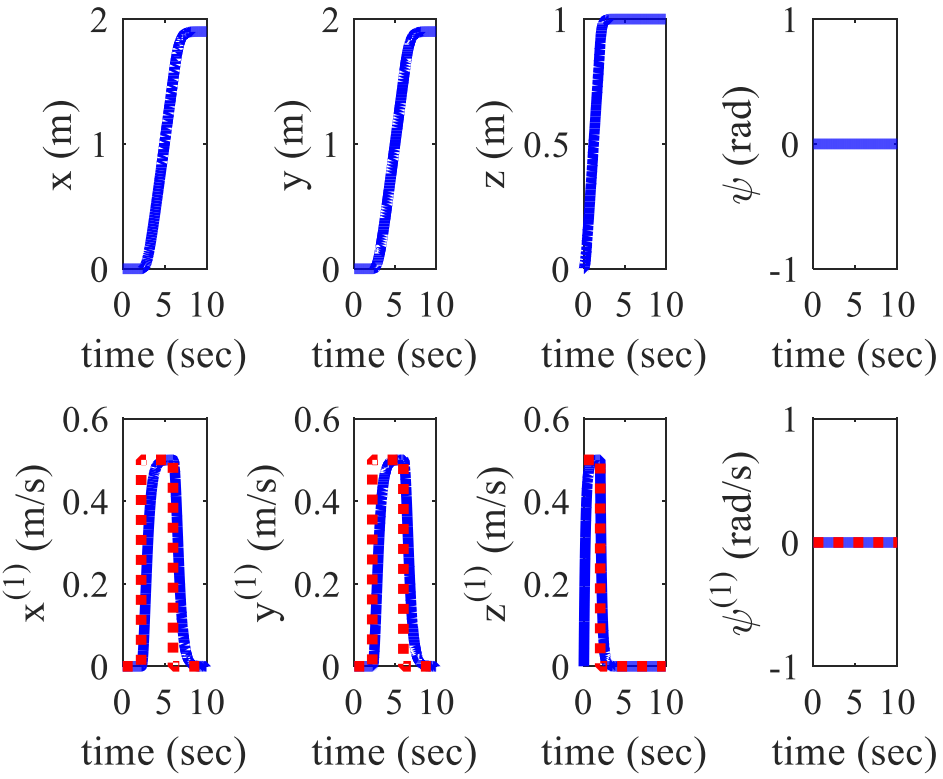
- $K_P = 8$.

Odabirom polova za regulator X i Y osi, a_1=-3, a_2=-5, a_3=-4.

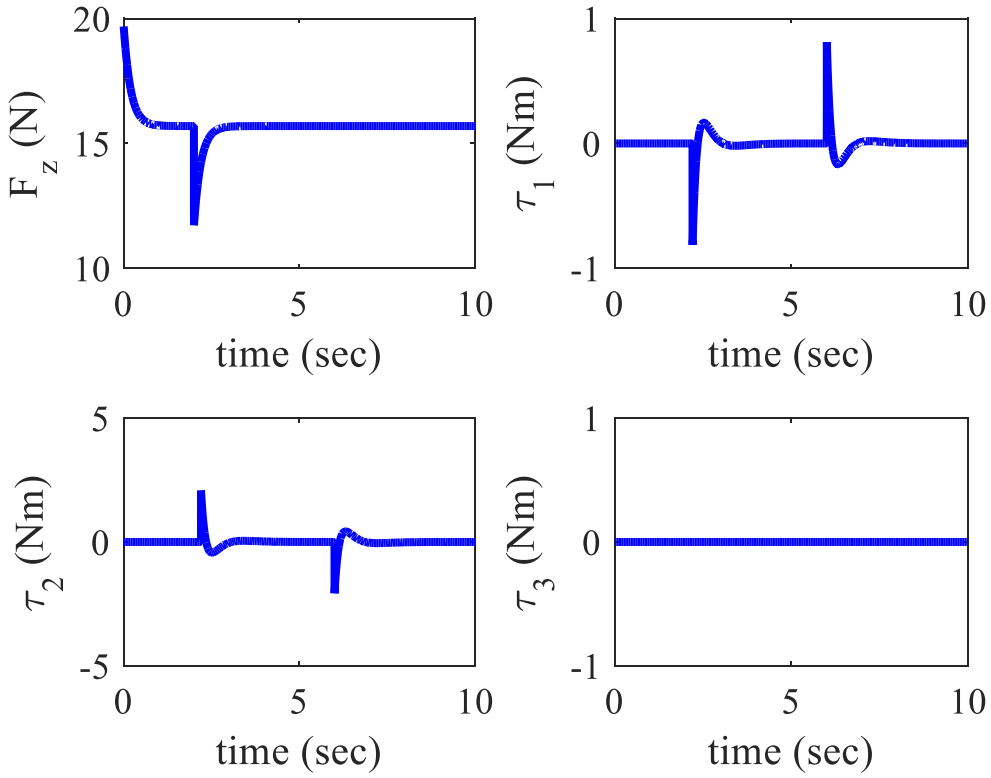
- K_0= 60, K_1= 47, K_2= 12.

4.2.1.1. Model 3

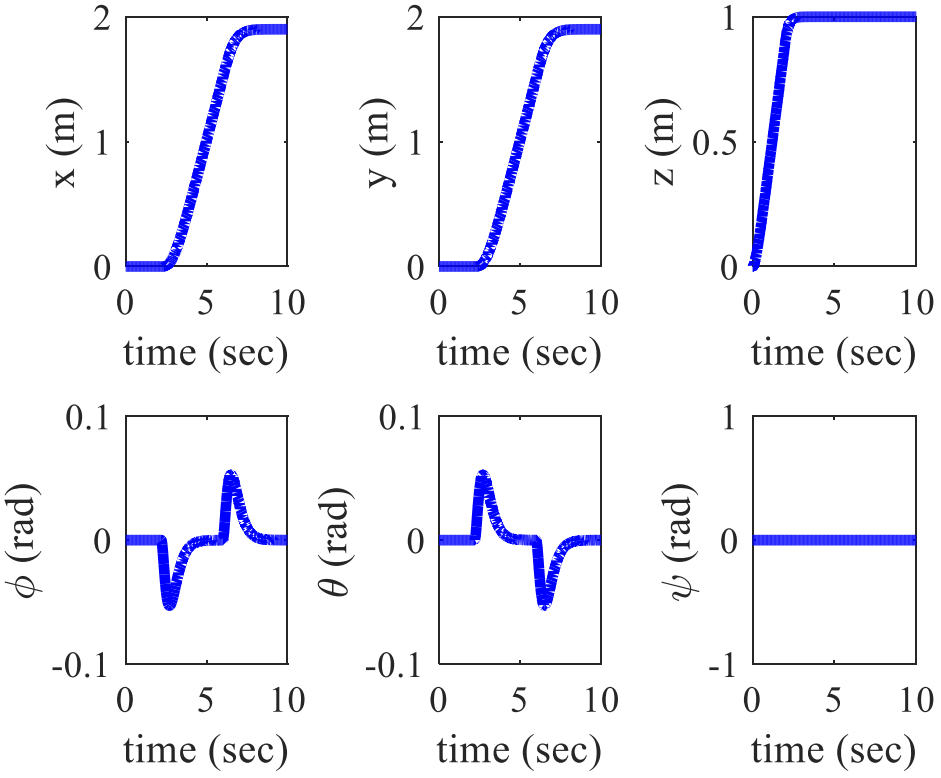
Slike 31 -33 prikazuju rezultate simulacije na linearizirranom modelu uz gore navedena pojačanja.



Slika 28 Odziv regulatora na zadane reference

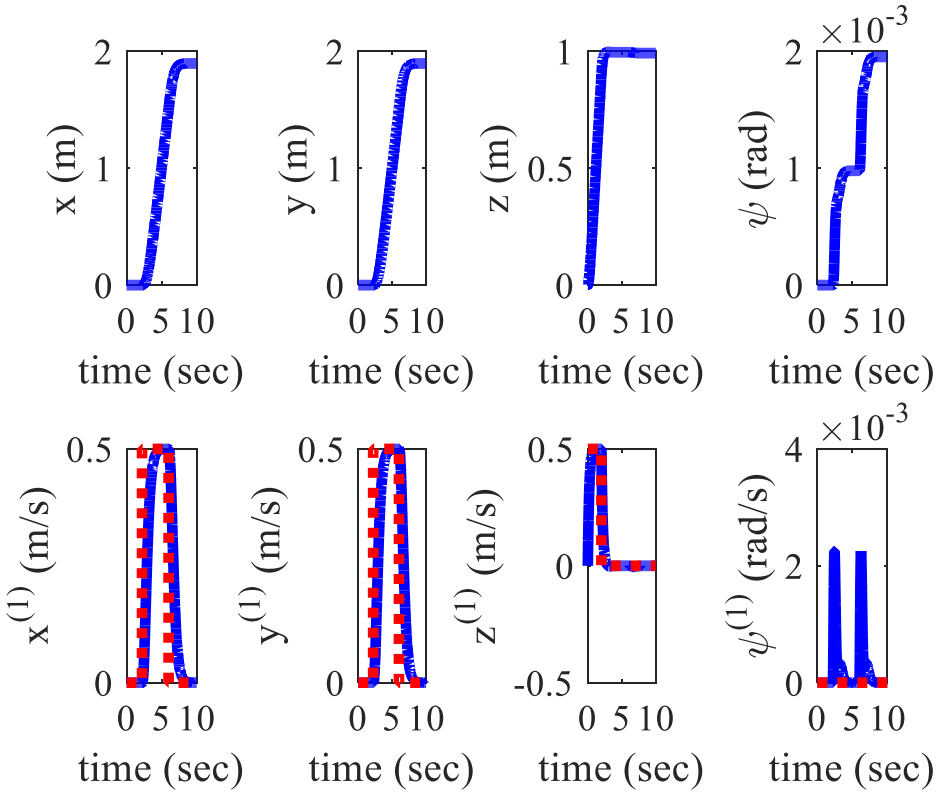


Slika 29 Upravljačke variable

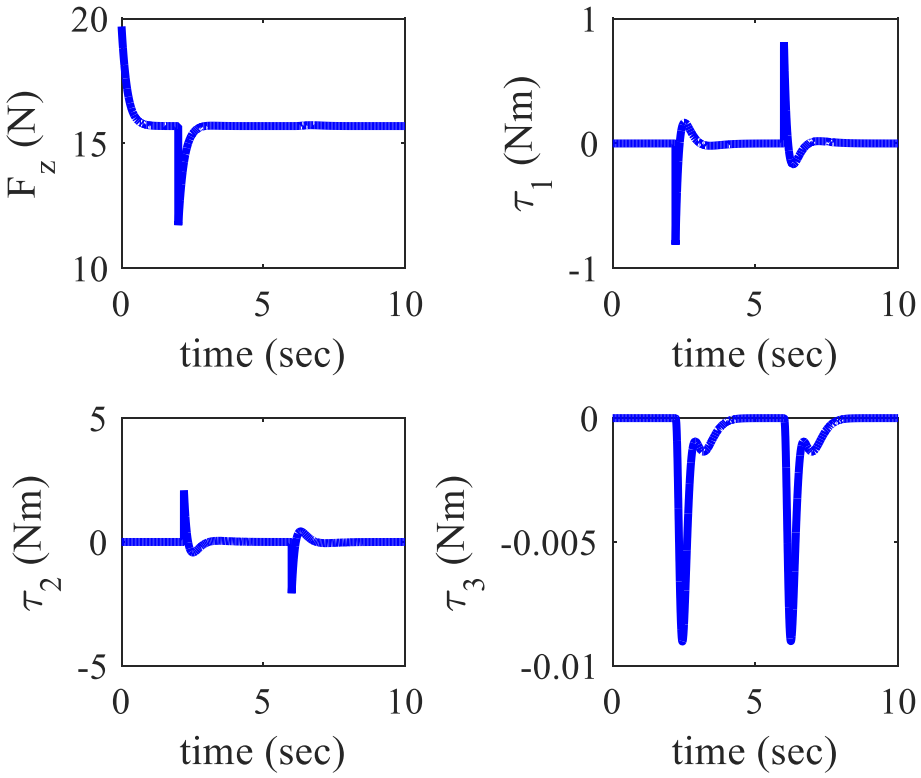


Slika 30 Pozicija i orijentacija tijela

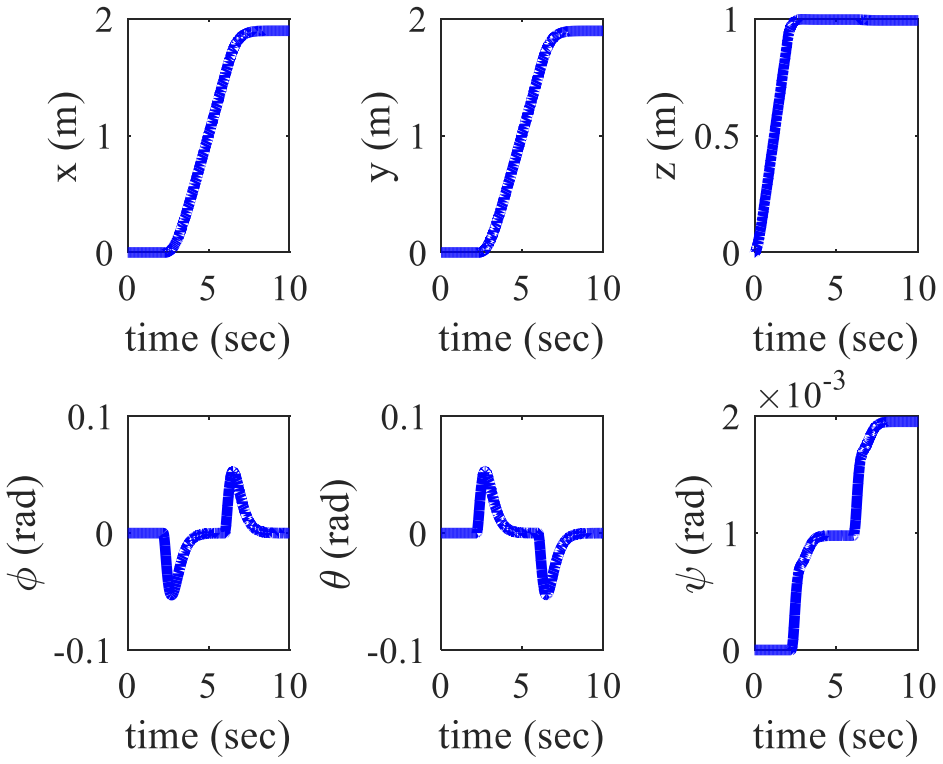
4.2.1.2. Model 2



Slika 31 Odziv regulatora na zadane reference



Slika 32 Upravljačke variable

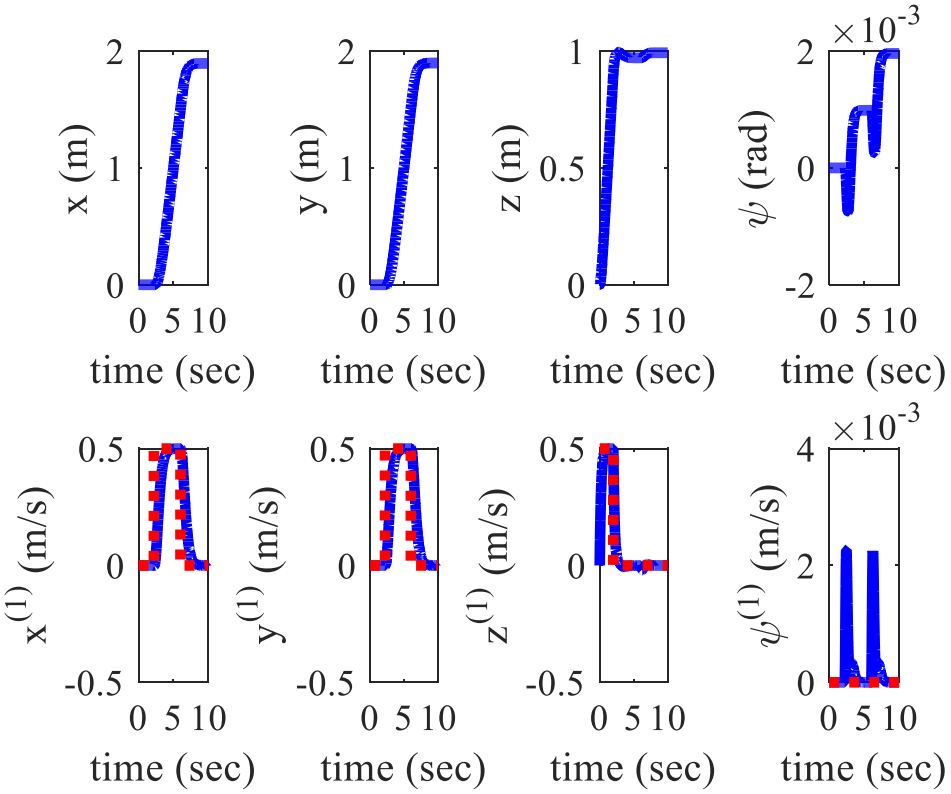


Slika 33 Pozicija i orijentacija tijela

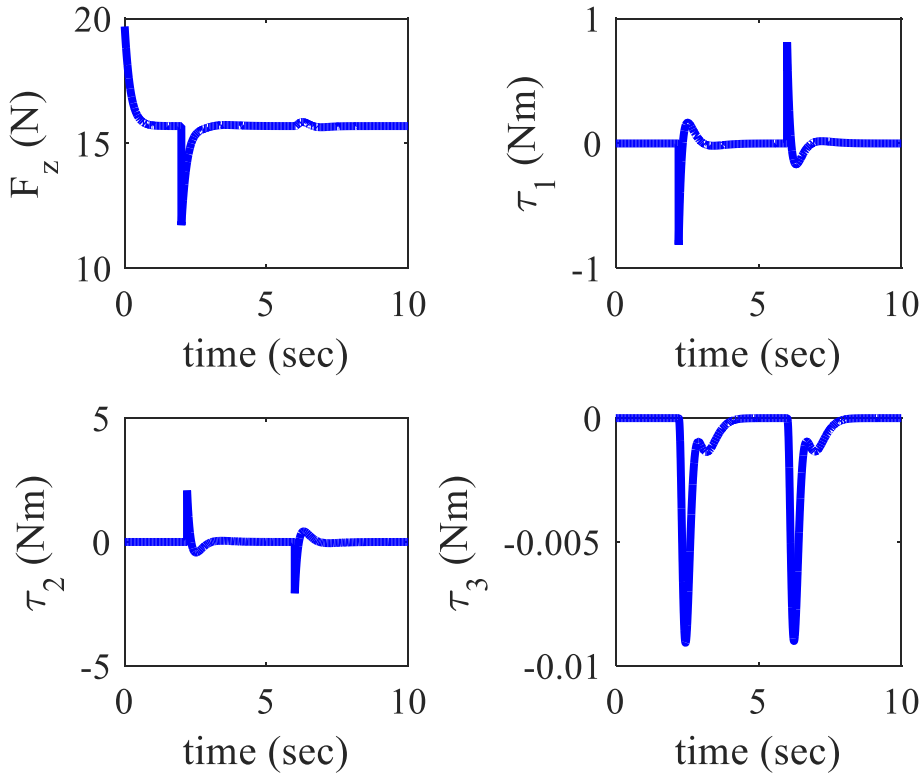
Slike 34 – 36 prikazuju rezultate simulacije regulatora po brzini za prvi set pojačanja na pojednostavljenom dinamičkom modelu.

4.2.1.3. Model 1

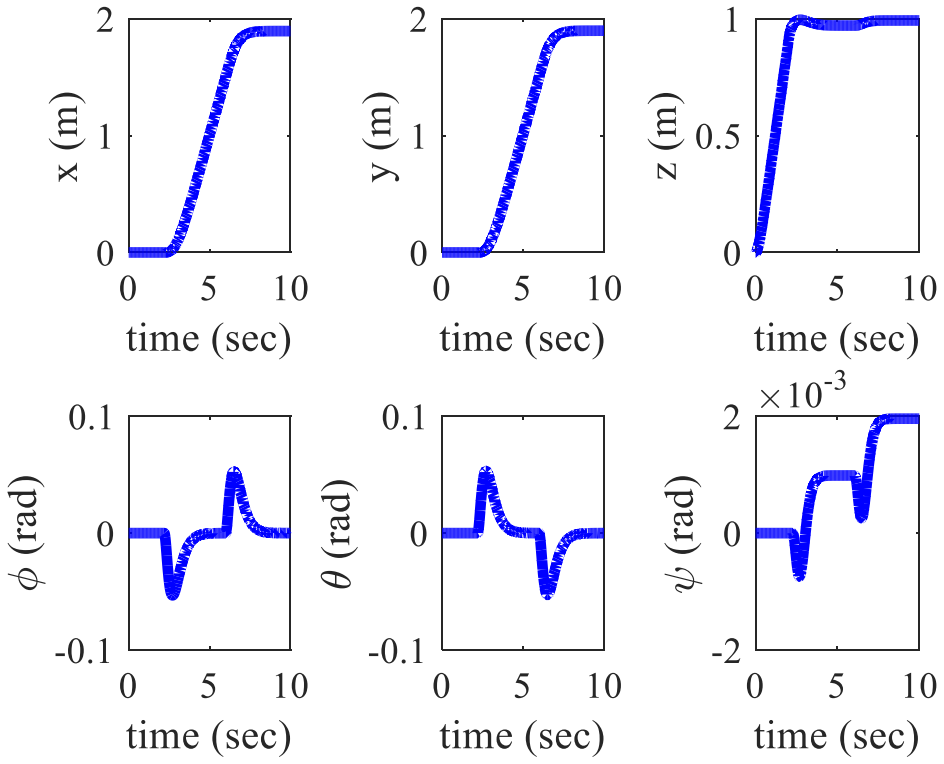
Slike 37 – 39 prikazuju rezultate simulacije na potpunom dinamičkom modelu.



Slika 34 Odziv regulatora na zadane reference



Slika 35 Upravljačke variable



Slika 36 Pozicija i orijentacija tijela

4.2.2. Drugi set pojačanja

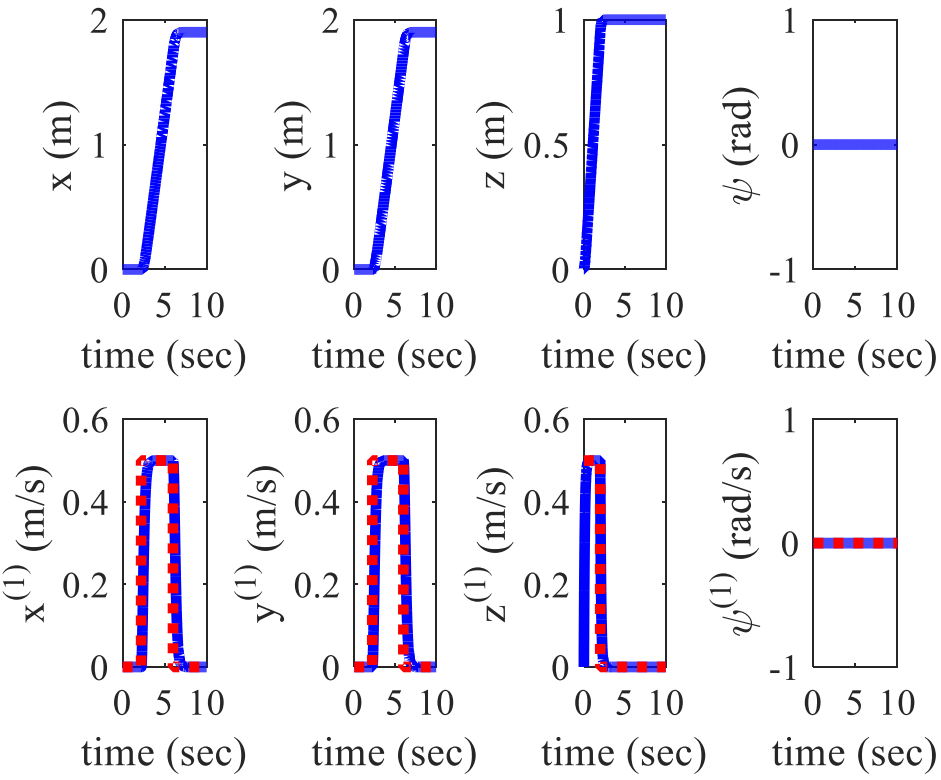
Pojačanje regulatora u smjeru osi Z je

- $K_P = 15$.

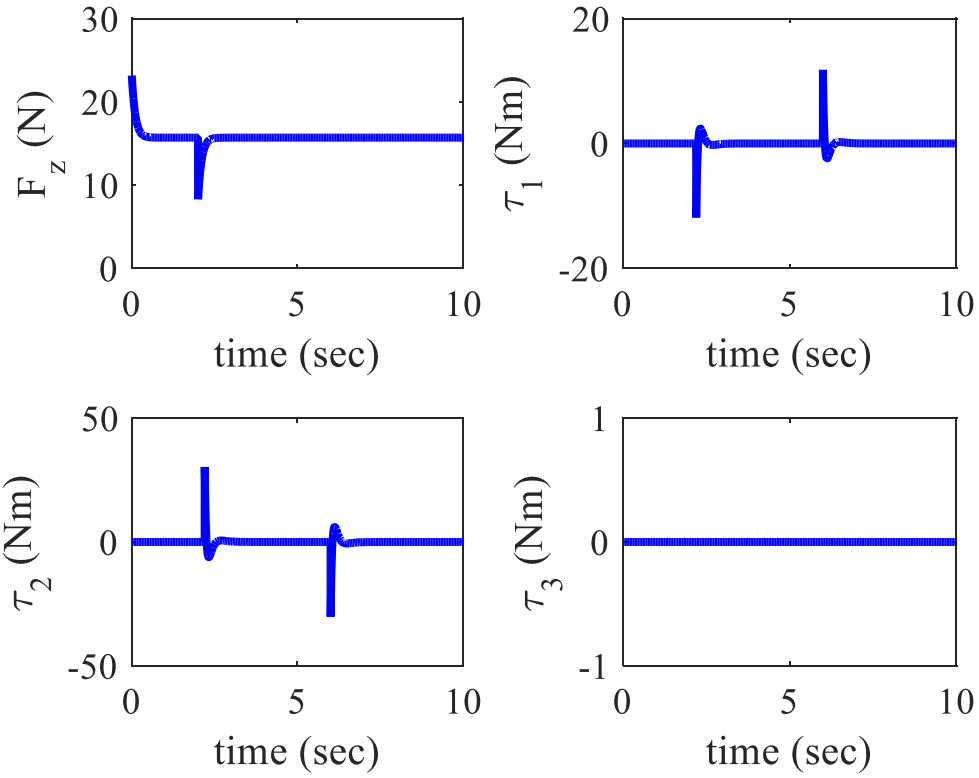
Odabirom polova za regulator X i Y osi, a1=-6, a2=-10, a3=-15.

- K0= 900, K1= 300, K2= 31.

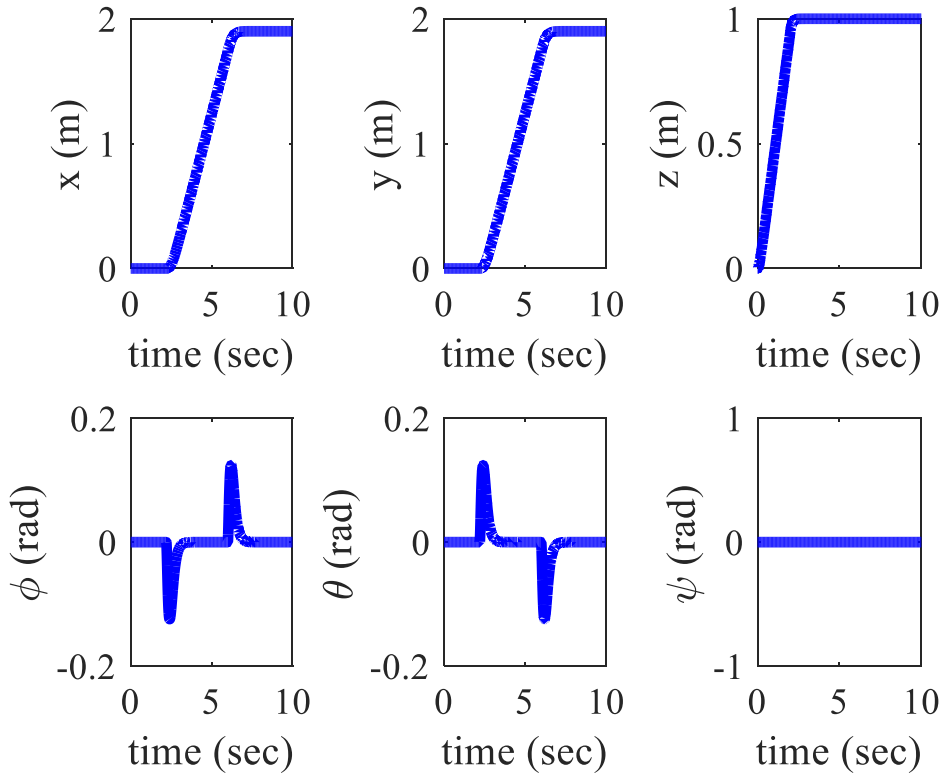
4.2.2.1. Model 3



Slika 37 Odziv regulatora na zadane reference



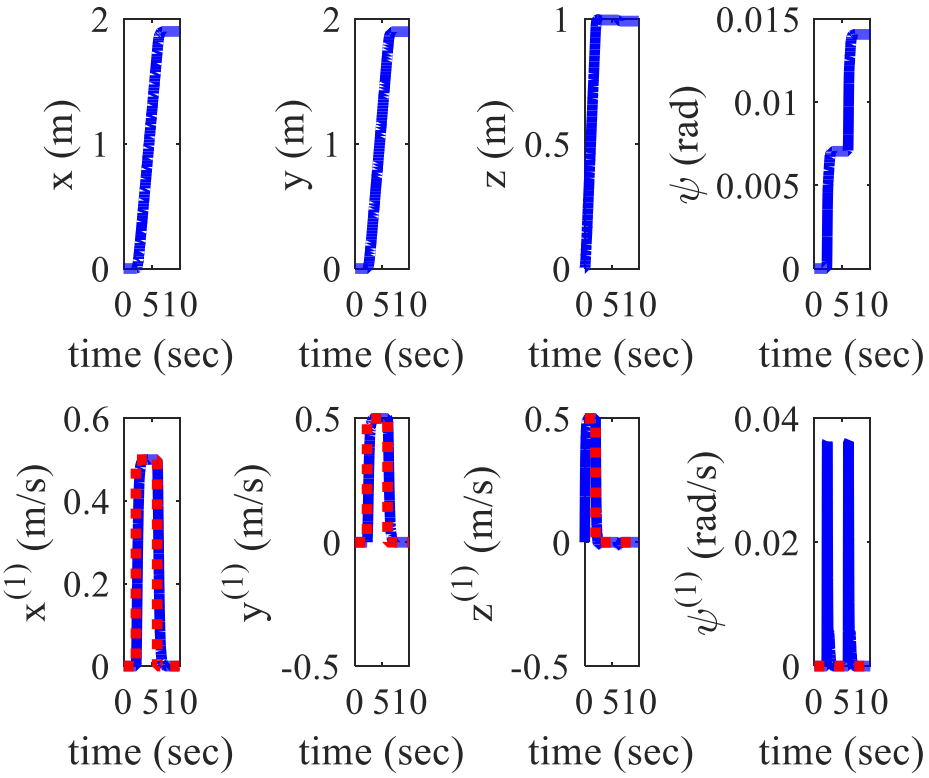
Slika 38 Upravljačke variable



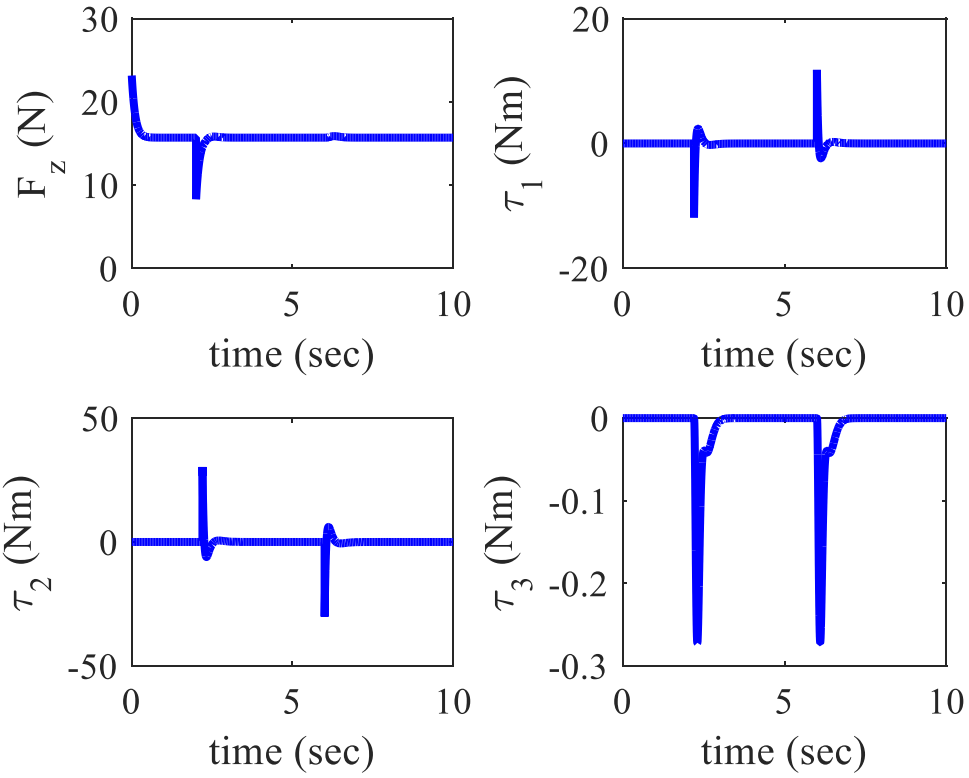
Slika 39 Pozicija i orijentacija tijela

Slike 40 – 42 prikazuju rezultate simulacije na modelu 3, potpuno lineariziranom, kao i očekivano zbog velikih pojačanja brzo se dostiže referentno stanje.

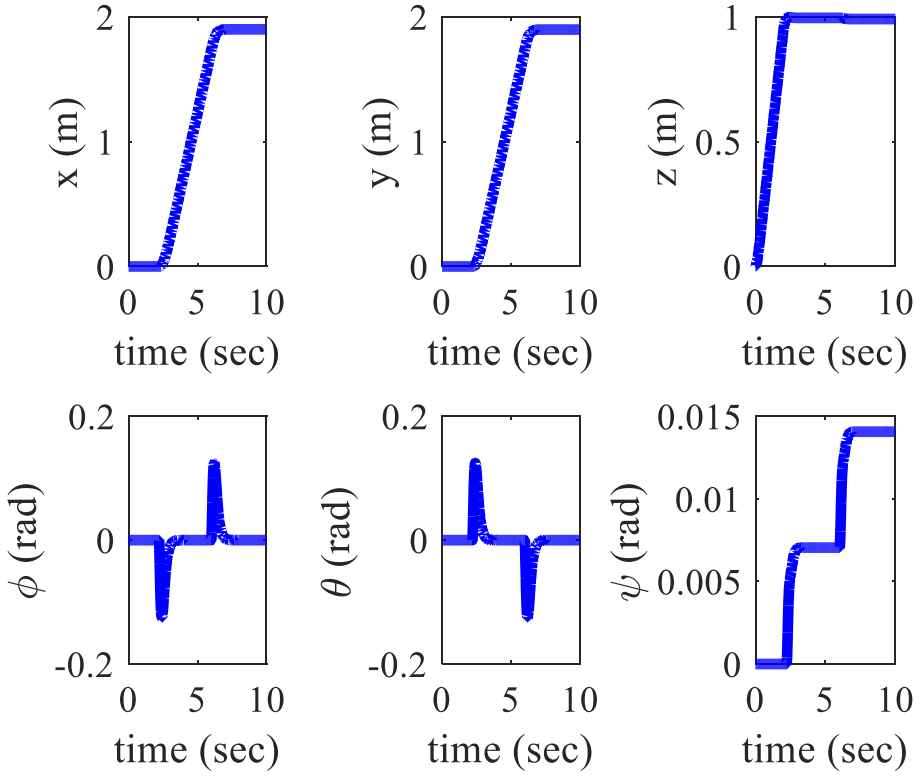
4.2.2.2. Model 2



Slika 40 Odziv regulatora na zadane reference



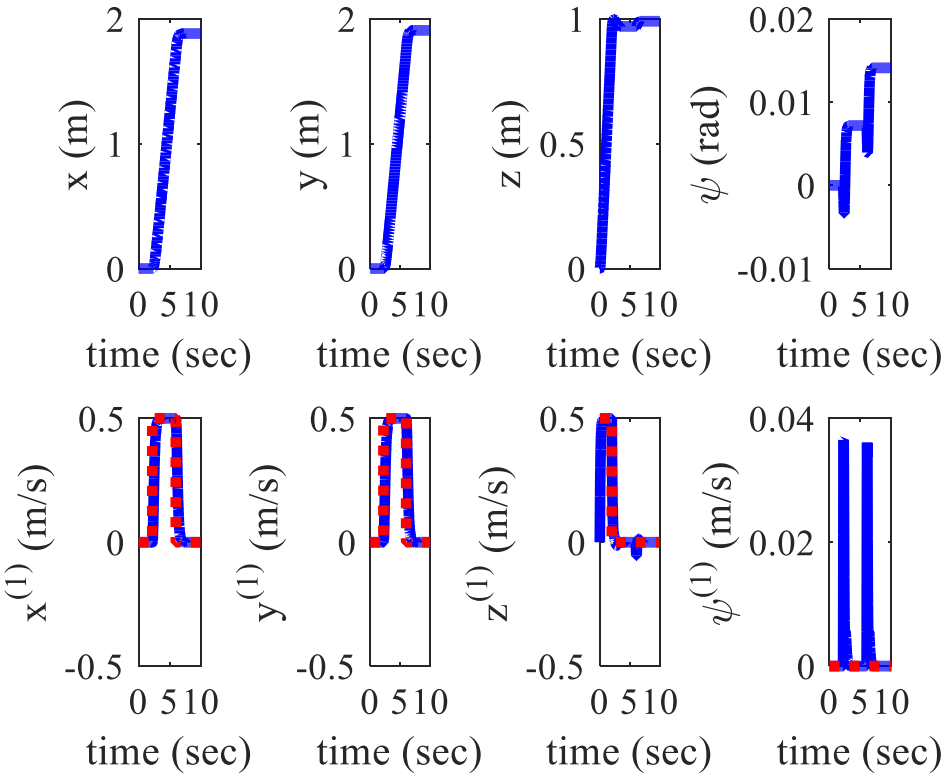
Slika 41 Upravljačke variable



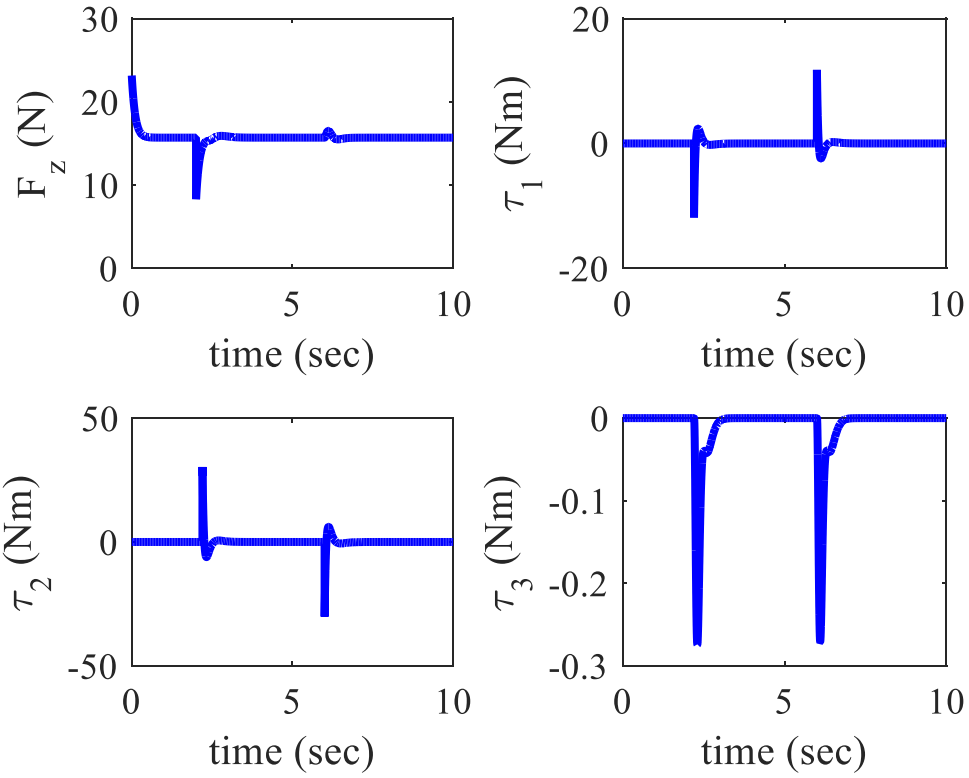
Slika 42 Pozicija i orijentacija tijela

Slike 43 – 45 prikazuju rezultate simulacije na pojednostavljenom modelu. Odziv je manji od 1s i nema prebačaja.

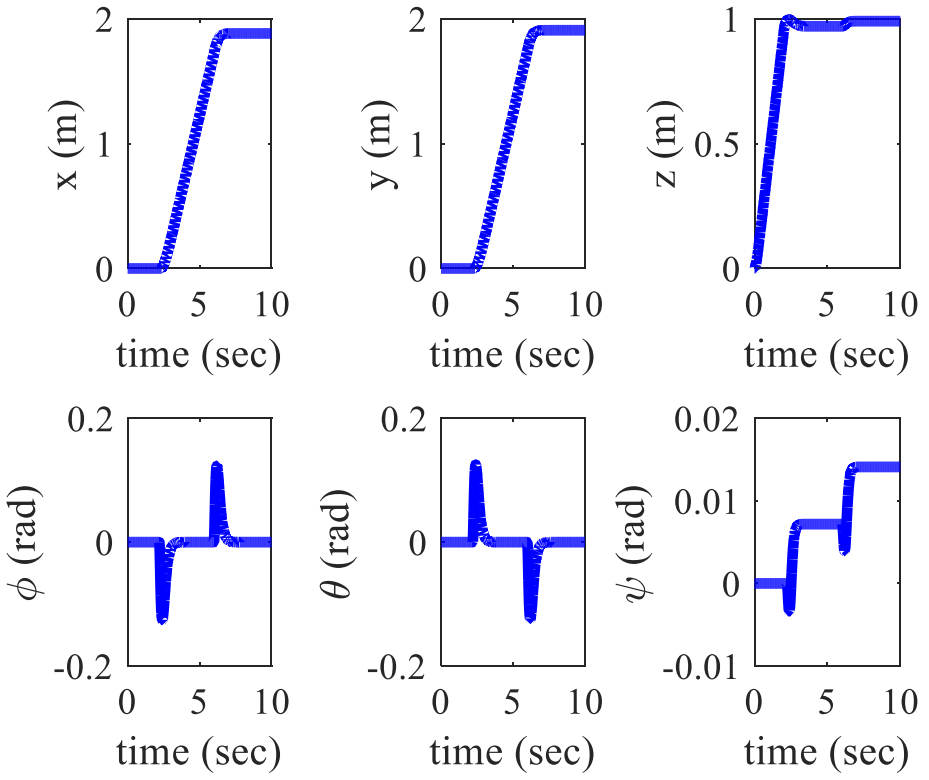
4.2.2.3. Model 1



Slika 43 Odziv regulatora na zadane reference



Slika 44 Upravljačke variable



Slika 45 Pozicija i orijentacija tijela

Slike 46 -48 pokazuju rezultate na punom dinamičkom modelu.

Za primjenu je odabran prvi set pojačanja, kako nebi bilo velikih opterećenja na aktuatore.

5. Praktični dio rada

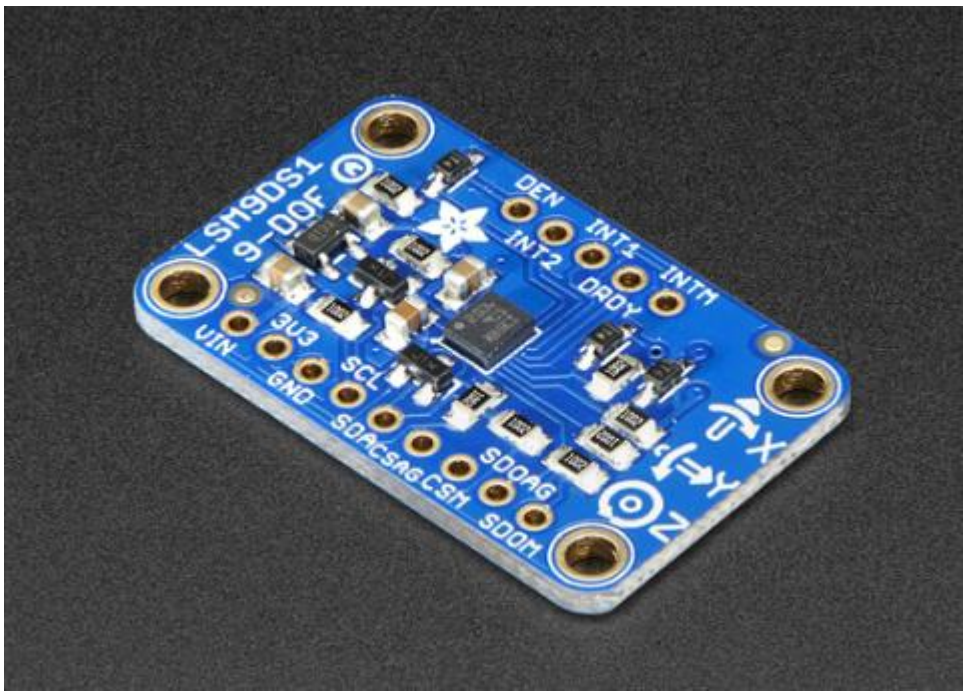
Za drona sa slike 2 bilo je potrebno odabrati elektroniku za prikupljanje podataka, obradu istih i upravljanje motorima. U taj sustav su implementirani upravljački algoritmi koji su prije u radu objašnjeni i ispitani u simulaciji.

Dron se sastoji od pet funkcionalne cjeline:

- Mehanički sustav – kućište koje drži sve ostale sustave,
- Aktuatori – motori i propeleri, u našem slučaju BLDC motori, SunnySky x2216, 880KV, Gemfan 11x4.7 propeleri,
- Energetski sustav – izvor električne energije, LiPo batterija 3S,ESC turniggy pulsh 30A,
- Mjerni sustav – senzori za mjerenje pozicije i orijentacije,
- Upravljački sustav – kontroler koji obrađuje podatke sa senzora i računa upravljačke variable.

U ovom radu obrađena su posljednja 2, mjerni i upravljački. Bitno je napomenuti i neke pojedinosti elemenata iz prve 3 skupine.

5.1. Senzori



Slika 46 Adafruit LSM9DS1

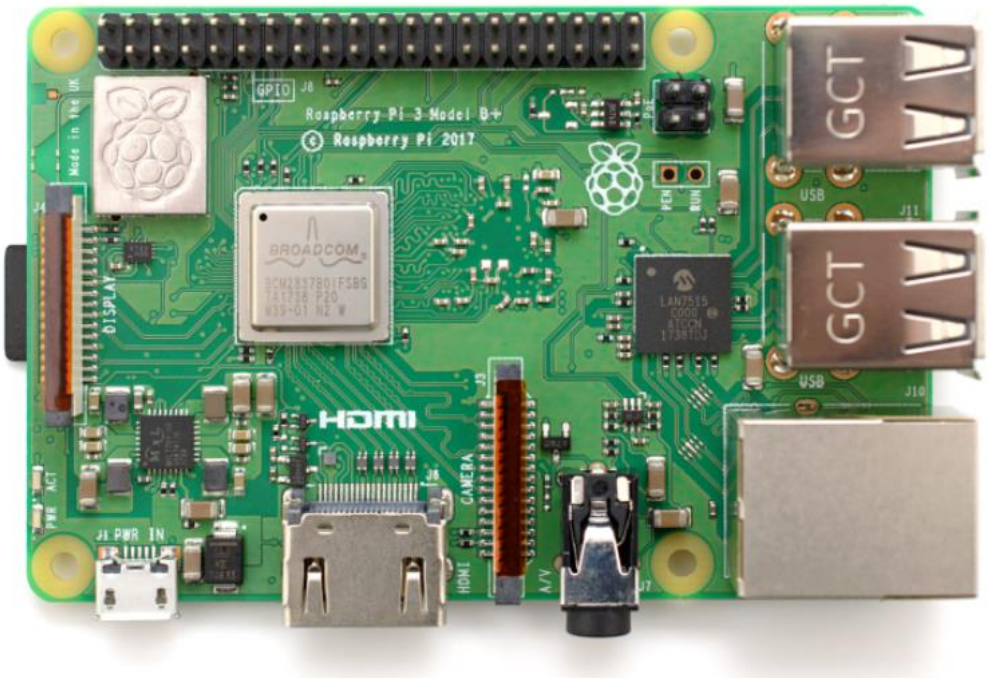
Odabran je LSM9DS1 čip, s 3 senzora u sebi, akcelerometar, žiroskop i magnetometar, u MEMS tehnologiji. Male dimenzije i svi senzori potrebni za navigiranje u zatvorenom, čine ga pogodnim za ovu primjenu. Svaki od senzora je tro osni, što znači da možemo mjeriti akceleraciju u svim smjerovima, kao i kutne brzine oko sve tri osi. Senzor je u kućištu postavljen tako da se njegove osi podudaraju s glavnim osima konstrukcije tako da njegova mjerenja možemo koristit uz vrlo malo obrade. Jedino što je potrebno napraviti je od mjerenih vrijednosti oduzeti Bias svake od njih, to je pogreška senzora koja uvijek postoji. Određuje se aritmetičkom sredinom, većeg broja mjerenja. Kako je prije u radu napomenuto, pošto smo model linearizirali oko stanja lebdenja, možemo direktno koristit mjerenja s žiroskopa, kao kutne brzine oko osi inercijskog okvira iako mjerenje vršimo i okviru vezanom uz letjelicu.

Tehnički podaci:

- Linearnu akceleraciju možemo mjeriti u više raspona, ±2/±4/±8/±16 *g*,
- Magnetsko polje ±4/±8/±12/±16 gauss,
- Kutnu brzinu ±245/±500/±2000 dps,
- 16-bit podatkovni izlaz,
- SPI/I^2C komunikacija,
- Napajanje 1.9V – 3.6V…

Za čitanje podataka iz senzora koristio se gotovu biblioteku s GitHuba, javni repozitorij. U librariju su podaci o registrima, adrese i neke funkcije, senzoru se pristupa kako je već navedeno I^2C sabirnicom što Raspberry može po defaultu sam zbog hardvarskog sučelja, obavlja se samo čitanje iz registara akcelerometra i žiroskopa.

5.2. Kontroler



Slika 47 Raspberry Pi 3B+

Kao glavni mozak ove operacije odabran je Raspberry PI 3B+, malo računalo razvijeno u Engleskoj. Veoma popularno računalo koje se koristi u svakakvim projektima. Njegova popularnost ujedno znači i brz razvoj novijih modela, kao i veliku zajednicu koja nudi puno gotovih rješenja, rješenja problema na koje će se naići i tokom ovog rada. Raspberry Pi dolazi s operativnim sustavom Raspbian, Linux distribucija, s kojim će se koristiti u radu. Postoji i real time Linux operativni sustav, ali nije implementiran u ovom radu. Raspberry Pi 3B+ vrlo je moćan, pokreče ga Broadcom BCM2837B0 quad-core A53 64-bit procesor koji radi na 1.4 GHz, ima 1 GB RAM-a, Micro-SD karticu do 128GB, mogućnost spajanja na Internet, žično i bežično. Jedna od najpovoljnijih stvari kod ovog računala je 40 GPIO pinova koji se mogu koristiti na razne načine, zato je i popularan kod puno ljudi. Na Raspberriju će se vrtjet program pisan u C-u, kako bi se što više dobilo na brzini. Raspberry pomoću I^2C sabirnice komunicira s ostalim sustavima, to je serijska komunikacija koja podržava više master-a i više slave-ova. I^2C je vrlo efikasan način komunikacije pošto koristi samo dvije žice, clock kojom master određuje kada se šalju bitovi i dana kojom putuju podaci. S radio prijamnikom je spoje preko serijske veze, s 115200 bauda ratom. U programu koji se vrti u beskonačnoj petlji, nakon inicijalizacije, u kojoj se provjerava da li su svi sustavi spojeni i rade li ispravno, postavlja izlaze u određeni položaj čime znamo da je letjelica spremna letjeti (motori se ne vrte). Poziva se funkcija za čitanje sa serije, na koju je spojen prijamnik (dobivamo referentne vrijednosti). Ako je zadovolje uvjet da je određeni kanal u visokom stanju ulazi se u petlju

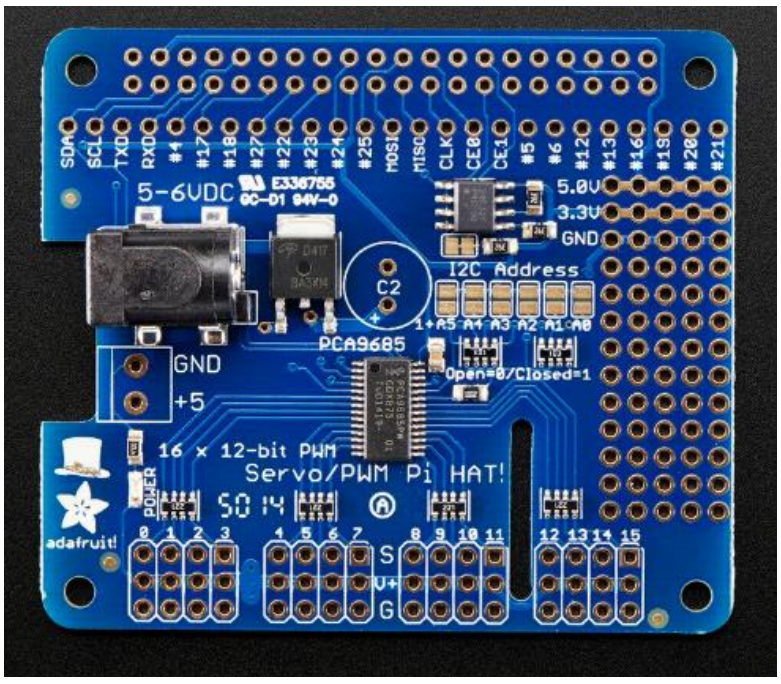
koja vrši upravljanje. U toj petlji čitaju se podaci s senzora (I^2C), za sada se samo akceleracije integriraju kako bi se dobila brzina što nije idealno zbog drifta, a kutevi se određuju komplementarnim filtrom u kojem se podaci s žiroskopa i akcelerometra spregnu zajedno. Nakon toga se računaju upravljačke veličine (3.36). Iz izračunatih upravljačkih veličina određuju se sile na pojedinom aktuatoru uz pomoć matrice (2.7) odnosno njenog inverza, te se generira PWM signal.

5.3. ESC

Electroni speed controller je elektronički sklop koji služi za kontrolu brzine vrtnje Brushless DC motora. Odabrani ESC ima na sebi BEC, battery eliminator circle, koji služi za napajanje niskoenergetskih krugova u sustavu. Svaki motor treba imati ESC, tako da ih ima četiri i svaki od njih napaja po jednu komponentu, senzor, Raspberry, radio prijamnik i PCA9685. ESC funkcionira tako da brzinu vrtnje motora kontrolira u ovisnosti o PWM signalu koji dobiva na ulazu, iz kontrolera. PWM je pulse width modulation, što bi značilo da širina pulsa nosi informaciju, vremenska duljina signala u visokom stanju. Osim informacija ovim načinom se može i regulirati napon na izlazu. ESC prima PWM signal od 50Hz što daje periodu od 20ms. Signal, u visokom stanju, od 1ms tumači se kao 0 gasa na motoru, a signal širine 2ms se tumači kao maximalni gas.

5.4. PCA9685

Odabrani čip nalazi se na tiskanoj pločici proizvođača adafruit.



Slika 48 Adafruit 16Ch, 12bit servo/pwm hat

Ova pločica kompenzira nedostatak Raspberry-ja da generira dovoljan broj PWM signal. Pločica je bazirana na odabranom čipu koji nam omogućuje istovremenu kontrolu do 16 izlaznih kanala. Kao i senzor koristi I^2C komunikaciju. Preko I^2C možemo adresirati registre u čipu i podesiti frekvenciju, 50Hz zato što takav signal traže ESC-ovi, pristupom drugim registrima tijekom leta mijenjamo PWM koji se generira za pojedini motor. Glavni program na Raspberry-ju računa koju bi silu motori trebali ostvariti i na temelju podataka koje je dao proizvođač generira se duljina PWM signala. Ovisno o potrebnoj duljini signala pune se registri PCA9685 čipa i on radi PWM izlaze. U programu se koristi library korisnika Reinbert S GitHuba, kao i za senzor to je skup podataka, najviše adresa raznih registara. I^2C- om šaljemo podatke u određene registre koji su zaduženi za svaki od izlaza, svaki izlaz ima 2 registra, šaljemo 12 bitni podatak kojim određujemo kada signal ide u visoko i kad u nisko stanje.

5.5. Turnigy Evolution radio transmiter i iA6c radio reciever

Turnigy Evolutio radio transmiter je 8 kanalni radio sustav, 2.4GHz. Rezolucija upravljača je 4096. Služi se za slanje referenci s zemlje. Središnji prekidač pali/gasi mogućnost letenja, u donjem položaju, program ne ulazi u petlju za let.

iA6c je prijamnik uparen s gore navedenim odašiljačem, domet je 300m. Za komunikaciju s kontrolerom leta može se koristiti više vrsti komunikacije PWM, PPM, s.bus ili i.bus. U ovom projektu odabran je i.bus protokol zbog svoje brzine i jednostavnosti. Serijska komunikacija 115200 baud rate, signal se šalje svakih 7ms. Vrijednosti svakog signala su od 1000-2000. Signal se obrađuje kako bi se dobila referentna vrijednost, u našem slučaju ±0.5m/s.

Slika 49 Turnigy Evolution i receiver

6. ZAKLJUČAK

Upravljački algoritmi su dovoljno robusni to se vidi na njihovim odzivima i na potpunom dinamičkom modelu kao i na potpuno linearizirano. Svakako bi trebalo implementirati PID regulator kako bi se regulacijska greška uklonila, no ni to ne bi bilo dovoljno. Trebalo bi razviti kompleksnije regulatore kako bi se mogli kompenzirati i vanjska djelovanja, poput udara vjetra.

Što se tiče praktičnog dijela, letjelica nije uspjela poletjeti, Linux koji je na raspberriju nije idealna podloga za upravljanje ovakvim sustavima, barem ne za nekoga tko se ne bavi linuxom i programiranjem. Trebalo bi pokušati sa real time linux-om ili s nekim drugim kontrolerom. Ostaje puno mjesta za danji rad na ovoj letjelici što je i bio cilj, napraviti nešto na čemu se može nastaviti istraživati i učiti.

Postoje već gotovi kontroleri leta kao što je Pixhawk. Jednostavnije rješenje, kontroler koji u sebi sadrži sve potrebno za let, senzore kakvi su i prije navedeni, pokreče ga već dobro uređen software, pošto je ovo 3 generacija tog kontrolera.

Slika 50 Pixhawk

LITERATURA

[1] Bresciani, T. (2008.), Modelling, Identification and Control of a Quadrotor Helicopter, Italy, Lund University, 2008

[2] Brezak H. (2017.), Robusno upravljanje autonomnom letjelicom s četiri rotora, Fakultet strojarstva i brodogradnje, Zagreb, 2017.

[3] Kasać, J., Stevanović, S., Žilić, T., Stepanić, J.(2013.), Robust Output Tracking Control of a Quadrotor in the Presence of External Disturbances, Zagreb, FAMENA, 2013.

[4] Nonami, K., Kendoul, F., Suzuki, S., Wang, W., Nakazawa D. (2010.), Autonomous Flying Robots, Japan, Springer, 2010.

[5] https://www.raspberrypi.org/forums/

[6] https://github.com/Reinbert/pca9685

[7] https://github.com/justinnuwin/liblsm9dsx

PRILOZI

I. CD-R disc

II. Matlab kod

III. C kod

C kod

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <wiringSerial.h>
#include <wiringPi.h>
#include <pca9685.h>
#include <lsm9ds1.h>
#include <time.h>
#include <math.h>

#define IBUS_BUFFERSIZE 32 //Max iBus paket s recivera
#define IBUS_MAXCHANNELS 7
#define KP 8
#define KK0 60
#define KK1 47
#define KK2 12

//gcc c.c -o program -Wall -lwiringPi -llsm9ds1 -lwiringPiPca9685 -llrt
static int ibusIndex = 0;
static char ibus[IBUS_BUFFERSIZE] = {0};
static short rcValue[IBUS_MAXCHANNELS];
static float ref[4];
static int rxFrameDone;
long double t;
int fds, fd, i, j = 0;
float c, b;
float acc[3] = {0};
float gyr[3] = {0};
float u1, u2, u3, u4;
long int tpocetak;
struct timespec gettime_now;
long int time_senzor;

void readRx(); //fun za citanje sa serijske
void min_max(float rp, float *prp);
//void rpm_PWM(float rpm1, float rpm2, float rpm3, float rpm4, int *p1, int *p2, int *p3, int *p4);
void uprav_u_PWM(float u1, float u2, float u3, float u4, int *r1, int *r2, int *r3, int *r4);
void chill();

int main()
{
	float Phi = 0, Theta = 0, Psi = 0;
	//		float bax=0.0474, bay=-0.1295, baz=0.3044, bgx=1.2099, bgy=1.1932, bgz=0.2329; //biasi
senzora
	float vx = 0, vy = 0, vz = 0;
	//float rpm1, rpm2, rpm3, rpm4;
	float m = 1.6, g = 9.81, ix = 0.72, iy = 0.69;
	int pwm1, pwm2, pwm3, pwm4;
	long double dt_long_double;

	wiringPiSetup();
	init_imu();

	pinMode(0, OUTPUT);
	digitalWrite(0, LOW); //na WiringPi Pin 0-->LED CRVENA, ne radi serija
	pinMode(1, OUTPUT);
	digitalWrite(0, LOW); //   -||-        1-->LED ZUTA, ne radi PWMi2c
```

```
if ((fds = serialOpen("/dev/serial0", 115200)) < 0)
{
	fprintf(stderr, "ne dela serija: %s\n", strerror(errno));
	digitalWrite(0, HIGH);
	delay(500);
	digitalWrite(0, LOW);
	delay(500);
	return 1;
}
init_imu();
if ((fd = pca9685Setup(300, 0x40, 50)) < 0)
{
	fprintf(stderr, "i2c do pwm ne dela");
	digitalWrite(1, HIGH);
	delay(500);
	digitalWrite(1, LOW);
	delay(500);
}
clock_gettime(CLOCK_REALTIME, &gettime_now);
time_senzor = gettime_now.tv_nsec;

for (;;)
{
	readRx();
	if (rxFrameDone == 1 && rcValue[5] == 1000)
	{
		chill();
	}

	if (rxFrameDone == 1 && rcValue[5] == 2000)
	{ // clock_gettime(CLOCK_REALTIME,&gettime_now);
		// tpocetak=gettime_now.tv_nsec; //da vidim kolko traje
		for (i = 0; i < 4; i++)
		{ //Skaliranje ulaza s rc-a na referentne brzine +/- 0.5m/s(rad/s)
			ref[i] = ((float)(rcValue[i] - 1500)) / 1000;
		}
		get_imu_reading(acc, gyr); //citanje senzora

		clock_gettime(CLOCK_REALTIME, &gettime_now);

		dt_long_double = gettime_now.tv_nsec -time_senzor;

		if (dt_long_double < 0)
		{
			dt_long_double += 1000000000;
		}

	time_senzor = gettime_now.tv_nsec;

	t = dt_long_double / 1000000000;
	//printf("vrijeme u s %Lf\n", t);
		acc[0] =acc[0]-bax; acc[1]= acc[1]-bay; acc[2]= acc[2]-baz;
		gyr[0]= gyr[0]-bgx; gyr[1]=gyr[1]-bgy; gyr[2]=gyr[2]-bgz; //ako micem bias

	vx = vx + acc[0] * t;
	vy = vy + acc[1] * t;
	vz = vz + (acc[2]-9.9) * t;
				//integriranje akceleracija
	//printf("\n brzina po Z: %f",vz);
```

```
                Phi = 0.95 * (Phi + (gyr[0] * t * M_PI / 180)) + 0.15 * acc[1] / acc[3]; //complementarni za
kuteve
                Theta = 0.85 * (Theta + gyr[1] * t * M_PI / 180) + 0.15 * (-1) * acc[0] / sqrt(pow(acc[1], 2) +
pow(acc[2], 2));
                Psi = Psi + gyr[2] * t * M_PI / 180;
                //printf("kutevi: %f  ,%f,  %f",Phi,Theta,Psi);
                u1 = -KP * (vz - ref[2]) + m * g; //zakon upravljanja, U1,U2,U3,U4
                u2 = -ix / g * (-KK2 * (-g * gyr[0]) - KK1 * (-g * Phi) - KK0 * (vy - ref[0]));
                u3 = iy / g * (-KK2 * (g * gyr[1]) - KK1 * (g * Theta) - KK0 * (vx - ref[1]));
                u4 = -1 * (KP * (gyr[3] - ref[3]));
                //printf("\nupravljacke: U1_%f  ,U2_%f  ,U3%f  ,U4 %f",u1,u2,u3,u4);
                uprav_u_PWM(u1, u2, u3, u4, &pwm1, &pwm2, &pwm3, &pwm4); //racuna U u PWM
                //printf("\nRPMzeljeni: rpm1_%f  ,2_ %f  ,3  %f  ,4 %f",rpm1,rpm2,rpm3,rpm4);
                //rpm_PWM(rpm1, rpm2, rpm3, rpm4, &pwm1, &pwm2, &pwm3, &pwm4);

        //      printf("\npwm za ibus %d %d %d %d", pwm1, pwm2, pwm3, pwm4);
                pca9685PWMWrite(fd, 0, 0, 205 + pwm1); //iBus output za motore
                pca9685PWMWrite(fd, 1, 0, 205 + pwm2);
                pca9685PWMWrite(fd, 2, 0, 205 + pwm3);
                pca9685PWMWrite(fd, 3, 0, 205 + pwm4);
                //clock_gettime(CLOCK_REALTIME,&gettime_now);
                //printf("u1,u2,u3,u4 %f,%f,%f,%f\n#f1= %f\n %f\n %f\n\n ",u1,u2,u3,u4,f1,c,b );
                //fflush(stdout);
                }
        }
}

//FUNKCIJE
void min_max(float rp, float *prp)
{
        if (rp <= 1)
        {
                *prp = 1;
        }
        if (rp >= 10)
        {
                *prp = 10;
        }
        else
        {
                *prp = rp;
        }
}
/*void rpm_PWM(float rpm1, float rpm2, float rpm3, float rpm4, int *p1, int *p2, int *p3, int *p4)
{
        min_max(rpm1, &rpm1);
        *p1 = (int)((rpm1 - 100) * 4.2);
        *p2 = (int)((rpm2 - 100) * 4.2);
        *p3 = (int)((rpm3 - 100) * 4.2);
        *p4 = (int)((rpm4 - 100) * 4.2);
}*/

void uprav_u_PWM(float u1, float u2, float u3, float u4, int *r1, int *r2, int *r3, int *r4)
{
        float f1, f2, f3, f4;
        //double fc1, fc2, fc3, fc4;
        f1 = 0.25 * u1 - 0.0016 * u3 + 2.5 * u4;
        f2 = 0.25 * u1 - 0.0016 * u2 - 2.5 * u4;
        f3 = 0.25 * u1 + 0.0016 * u3 + 2.5 * u4;
        f4 = 0.25 * u1 + 0.0016 * u2 - 2.5 * u4;
```

```
        min_max(f1,&f1);
    min_max(f2,&f2);
    min_max(f3,&f3);
    min_max(f4,&f4);
        *r1 = f1*20.4;
    *r2 = f2*20.4;
    *r3 = f3*20.4;
    *r4 = f4*20.4;

        /*fc1 = f1 /8 * 1000000;
        fc2 = f2 /8 * 1000000;
        fc3 = f3 /8* 1000000;
        fc4 = f4 /8 * 1000000;
         printf("\nomeg21 %f\n", fc1); printf("omeg23 %f\n", fc3);
         printf("Sila3 %f\n", f3); printf("Sila1 %f\n", f1); printf("Sila2 %f\n", f2); printf("Sila4 %f\n", f4);
        *r1 = sqrt(fc1);
        *r2 = sqrt(fc2);
        *r3 = sqrt(fc3);
        *r4 = sqrt(fc4);*/
}

void chill()
{
        pca9685PWMWrite(fd, 0, 0, 2040);
        pca9685PWMWrite(fd, 1, 0, 2040);
        pca9685PWMWrite(fd, 2, 0, 2040);
        pca9685PWMWrite(fd, 3, 0, 2040);
        rxFrameDone = 0;
}

void readRx()
{
        rxFrameDone = 0;
        char val = serialGetchar(fds);
        if (ibusIndex == 0 && val != 0x20)
        {
                ibusIndex = 0;
                return;
        }
        if (ibusIndex == 1 && val != 0x40)
        {
                ibusIndex = 0;
                return;
        }
        if (ibusIndex == IBUS_BUFFERSIZE)
        {
                rxFrameDone = 1;
                ibusIndex = 0;
                rcValue[0] = (ibus[3] << 8) + ibus[2];
                rcValue[1] = (ibus[5] << 8) + ibus[4];
                rcValue[2] = (ibus[7] << 8) + ibus[6];
                rcValue[3] = (ibus[9] << 8) + ibus[8];
                rcValue[4] = (ibus[11] << 8) + ibus[10];
                rcValue[5] = (ibus[13] << 8) + ibus[12];
                rcValue[6] = (ibus[15] << 8) + ibus[14];
        }
        if (val < 0)
        {
                printf("neprimam");
        }
```

```
        else
        {
                ibus[ibusIndex] = val;
                ibusIndex++;
        }
}
```

Matlab modeli

```
function dy = dron1(t,y)
dy = zeros(17,1);
global mm Ixx Iyy Izz id_zz g Vx0 Ay0 k0 k1 k2 k3 Kpz Kdz Kiz Kpw
global Kdw ky0 ky1 ky2 ky3 QQ r_g Sg I_B
if (QQ == 2)|(QQ == 3)
X=y(1); Xd=y(2);
Y=y(3); Yd=y(4);
Z=y(5); Zd=y(6);
Phi=y(7); Phid=y(8);
Theta=y(9); Thetad=y(10);
Psi=y(11); Psid=y(12);
end
if (QQ == 1)
%---State variables--MODEL 1-----------------------------------------%
% Skew-symmetrix matrix
S = @(x) [0, -x(3), x(2); x(3), 0, -x(1); -x(2), x(1), 0];
% Rotation matrix
Rx = @(x) [1, 0, 0; 0, cos(x), sin(x); 0, -sin(x), cos(x)];
Ry = @(x) [cos(x), 0, -sin(x); 0, 1, 0; sin(x), 0, cos(x)];
Rz = @(x) [cos(x), sin(x), 0; -sin(x), cos(x), 0; 0, 0, 1];
RR = @(x) Rz(x(3)).'*Ry(x(2)).'*Rx(x(1)).'; % x = [phi, theta, psi];
X=y(1); Xd=y(7);
Y=y(2); Yd=y(8);
Z=y(3); Zd=y(9);
Phi=y(4); Phid=y(10);
Theta=y(5); Thetad=y(11);
Psi=y(6); Psid=y(12);
% Variables:
Position=y(1:3);
Angle=y(4:6);
Velocity1=y(7:9);
Velocity2=y(10:12);
Velocity=[Velocity1; Velocity2];
%-------------------------------------------------------------------%
end
cPhi=cos(Phi); sPhi=sin(Phi);
cTheta=cos(Theta); sTheta=sin(Theta);
cPsi=cos(Psi); sPsi=sin(Psi);
%---Reference trajectory----------------------------------------%
z_d = Vx0*t;
x_d = -Ay0*0 + Ay0*cos(Vx0*t);
y_d = Ay0*sin(Vx0*t);
%Psi_d = Vx0*t; d1Psi_d = Vx0;
d1z_d = Vx0; d2z_d = 0; d3z_d = 0; d4z_d = 0;
d1x_d = -Ay0*Vx0*sin(Vx0*t); d2x_d = -Ay0*Vx0^2*cos(Vx0*t);
d3x_d = Ay0*Vx0^3*sin(Vx0*t); d4x_d = Ay0*Vx0^4*cos(Vx0*t);
d1y_d = Ay0*Vx0*cos(Vx0*t); d2y_d = -Ay0*Vx0^2*sin(Vx0*t);
d3y_d = -Ay0*Vx0^3*cos(Vx0*t); d4y_d = Ay0*Vx0^4*sin(Vx0*t);
% z_d=4;
% x_d=2;
% y_d=4;
Psi_d = 0;
% d1z_d = 0; d2z_d = 0; d3z_d = 0; d4z_d = 0;
```

```
% d1x_d = 0; d2x_d = 0; d3x_d = 0; d4x_d = 0;
% d1y_d = 0; d2y_d = 0; d3y_d = 0; d4y_d = 0;
d1Psi_d = 0;
d2Psi_d = 0;
%-------------------------------------------------------------%
%---Error variables-----------------------------------------%
d_z = Z-z_d; d1d_z = Zd-d1z_d;
d_x = X-x_d; d1d_x = Xd-d1x_d;
d_y = Y-y_d; d1d_y = Yd-d1y_d;
d_Psi = Psi-Psi_d; d1d_Psi = Psid-d1Psi_d;
%Parametri
m = 1.0*mm;
Ix = 1.0*Ixx;
Iy = 1.0*Iyy;
Iz = 1.0*Izz;
%stabilizacija Z,Psi, x,y
% U1 = d2z_d -Kpz*d_z -Kdz*d1d_z -Kiz*y(17);PID
U1 = m*(d2z_d -Kpz*d_z -Kdz*d1d_z);
U4 = Izz*(d2Psi_d -Kpw*d_Psi - Kdw*d1d_Psi);
d2d_x = g*Theta -d2x_d ;
d3d_x = g*Thetad -d3x_d ;
U3 = Iyy/g *(d4x_d -k3*d3d_x -k2*d2d_x -k1*d1d_x -k0*d_x);
d2d_y = -g*Phi -d2y_d ;
d3d_y = -g*Phid -d3y_d ;
U2 = -Ixx/g *(d4y_d -ky3*d3d_y -ky2*d2d_y -ky1*d1d_y -ky0*d_y);
%Model
if (QQ == 2)
% %---MODEL 2-------------------------------------------------------------%
dy(1) = y(2);
dy(2) = (cPhi*sTheta*cPsi + sPhi*sPsi)*(U1/mm);
dy(3) = y(4);
dy(4) = (cPhi*sTheta*sPsi - sPhi*cPsi)*(U1/mm);
dy(5) = y(6);
dy(6) = -g + (cPhi*cTheta)*(U1/mm);
dy(7) = y(8);
dy(8) = ((Iyy-Izz)/Ixx)*Thetad*Psid + (1/Ixx)*U2;
dy(9) = y(10);
dy(10) = ((Izz-Ixx)/Iyy)*Phid*Psid + (1/Iyy)*U3;
dy(11) = y(12);
dy(12) = ((Ixx-Iyy)/Izz)*Phid*Thetad + (1/Izz)*U4;
% %-------------------------------------------------------------%
end
if (QQ == 1)
% %---MODEL 1-------------------------------------------------------------%
e3=[0; 0; 1];
Forc=[0; 0; U1];
Torq=[U2; U3; U4];
U=1*[Forc; Torq];
c = cos(Angle); s = sin(Angle);
R_1 = (1/c(2))*[c(2), s(1)*s(2), c(1)*s(2);
0, c(1)*c(2), -s(1)*c(2);
0, s(1), c(1)];
M=[mm*eye(3), -mm*S(r_g); mm*S(r_g), I_B];
C=[mm*S(Velocity2), -mm*S(Velocity2)*S(r_g);
mm*S(r_g)*S(Velocity2), -S(I_B*Velocity2)];
G=-mm*g*[RR(Angle).'*e3; S(r_g)*RR(Angle).'*e3];
dy(1:3) = RR(Angle)*Velocity1; %Position=y(1:3);
dy(4:6) = R_1*Velocity2; %Angle=y(4:6);
dy(7:12) = inv(M)*(-C*Velocity + G + U);%Velocity1=y(7:9);Velocity2=y(10:12);
% %-------------------------------------------------------------%
end
if(QQ == 3)
dy(1) = Xd;
```

```
dy(2) = g * Theta;
dy(3) = Yd;
dy(4) = -g * Phi;
dy(5) = Zd;
dy(6) = (1/m)*(U1);
dy(7) = Phid;
dy(8) = (1/Ixx)*U2;
dy(9) = Thetad;
dy(10) = (1/Iyy)*U3;
dy(11) = Psid;
dy(12) = (1/Izz)*U4;
end
dy(13) = U1;
dy(14) = U2;
dy(15) = U3;
dy(16) = U4;
dy(17) = d_z;
```

Matlab- upravljanje po poziciji

```
clear all; close all;
clc
T=40; %simulation time
global mm Ixx Iyy Izz id_zz g Vx0 Ay0 k0 k1 k2 k3 Kpz Kdz
global Kiz Kpw Kdw ky0 ky1 ky2 ky3 QQ r_g Sg I_B
mm = 1.7;
Ixy = 0; Iyz = 0; Ixz = 0;
Vx0=0.5; Ay0=1;
g=9.81; x_g=0.0; y_g=0.0; z_g=0.0; r_g=[x_g; y_g; z_g];
Ixx = 0.27; Iyy = 0.62; Izz = 0.50; Ixy = 0; Iyz = 0; Ixz = 0; % momenti inercije
I_B = [Ixx -Ixy -Ixz; -Ixy Iyy -Iyz; -Ixz -Iyz Izz];
%QQ = 1; % MODEL 1 (full rigid-body dynamic model)
QQ = 2; % MODEL 2 (simplified rigid-body dynamic model)
%QQ = 3; % MODEL 3 (more simplified rigid-body dynamic model)
%Poja?anja Z, polovi
% a1= -2; a2= -2; a3 = -3;
% Kpz = a1*a2+a1*a3+a2*a3;
% Kdz = -(a1+a2+a3);
% Kiz = -a1*a2*a3;
%Poja?anja Psi, polovi
b1= -12; b2 = -20;
Kpw = b1*b2;
Kdw = -(b1+b2);
Kpz=Kpw
Kdz=Kdw
%Poja?anje x,y
c1 = -7; c2 = -8; c3 = -14; c4= -15;
k0 = c1*c2*c3*c4
k1 = -(c1*c2*(c3+c4)+c3*c4*(c1+c2))
k2 = c1*c2+c3*c4+(c1+c2)*(c3+c4)
k3 = -(c1+c2+c3+c4)
ky0 =k0;
ky1 =k1;
ky2 =k2;
ky3 =k3;
xx0=zeros(1,17);
if (QQ == 2)|(QQ == 3)
xx0(7)=0*0.1; xx0(9)=0*0.1; xx0(11)=0*0.1; % initial angles
end
if (QQ == 1)
xx0(4)=0*0.1; xx0(5)=0*0.1; xx0(6)=0*0.1; % initial angles
end
%xx0(17) = -0.95*mm*g/Kiz; % pocetno stanje integratora
%options = odeset('RelTol',1e-6,'AbsTol',1e-6);
```

```
%[t,y] = ode45('dron1',[0 T],xx0,options);
tspan=[0,T]; Nstep=10000; DelT = T/Nstep;
[t, y] = rk4(@dron1, tspan, xx0, DelT);
%-----------------------------------------
%---Reference trajectory-----------------------------------%
z_d = Vx0*t;
x_d = -Ay0*0 + Ay0*cos(Vx0*t);
y_d = Ay0*sin(Vx0*t);
% z_d=4;
% x_d=2;
% y_d=4;
Psi_d = 0;
if (QQ == 2)|(QQ == 3)
%------MODEL 2----MODEL 3-----------------------------------%
%----------------------------------------------------------%
figure(1)
subplot(2,3,1), plot(t,y(:,1),'b', t,x_d,'r:', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,2), plot(t,y(:,3),'b', t,y_d,'r:', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,3), plot(t,y(:,5),'b', t,z_d,'r:', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,4), plot(t,y(:,7),'b', 'linewidth',4), ylabel('\phi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,5), plot(t,y(:,9),'b', 'linewidth',4), ylabel('\theta (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,6), plot(t,y(:,11),'b', 'linewidth',4), ylabel('\psi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
%----------------------------------------------------------%
%---3D trajektorija-----------------------------------------%
figure(2)
plot3(y(:,1),y(:,3),y(:,5), 'b', x_d, y_d, z_d, 'r:', 'linewidth',3)
ylabel('x (m)','FontSize',16,'FontName','Times'), xlabel('y (m)','FontSize',
16,'FontName','Times'), zlabel('z (m)','FontSize',16,'FontName','Times'),
% axis([-1.1 1.1 -1.1 1.1 0 20])
set(gca,'fontsize',14,'FontName','Times'),
grid on
axis square
%----------------------------------------------------------%
%---Sila i momenti------------------------------------------%
F=diff(y(:,13))./diff(t);
M1=diff(y(:,14))./diff(t);
M2=diff(y(:,15))./diff(t);
M3=diff(y(:,16))./diff(t);
td=t(1:(length(t)-1));
figure(3)
subplot(2,2,1), plot(t,y(:,13),'b', 'linewidth',3), ylabel('F_z (N)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 9 15])
subplot(2,2,2), plot(t,y(:,14),'b', 'linewidth',3), ylabel('\tau_1 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -10 5])
subplot(2,2,3), plot(t,y(:,15),'b', 'linewidth',3), ylabel('\tau_2 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 5])
subplot(2,2,4), plot(t,y(:,16),'b', 'linewidth',3), ylabel('\tau_3 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
```

```
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 1])
% subplot(2,2,1), plot(td,F,'b', 'linewidth',3), ylabel('F_z (N)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 9 15])
% subplot(2,2,2), plot(td,M1,'b', 'linewidth',3), ylabel('\tau_1 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -10 5])
% subplot(2,2,3), plot(td,M2,'b', 'linewidth',3), ylabel('\tau_2 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 5])
% subplot(2,2,4), plot(td,M3,'b', 'linewidth',3), ylabel('\tau_3 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 1])
%-------------------------------------------------------------%
%---2D projekcije trajektorije--------------------------------%
figure(5)
subplot(2,1,1), plot(y(:,1),y(:,5),'b', x_d, z_d,'r:','linewidth',4), xlabel('x
(m)','FontSize',16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'),
%axis([-1.1 1.1 -1.1 1.1])
subplot(2,1,2), plot(y(:,3),y(:,5),'b', y_d, z_d,'r:', 'linewidth',4), xlabel('y
(m)','FontSize',16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'),
%axis([-1.1 1.1 0 20])
%-------------------------------------------------------------%
end
if (QQ == 1)
%------MODEL 1-----------------------------------------------------%
%-------------------------------------------------------------%
figure(1)
subplot(2,3,1), plot(t,y(:,1),'b', t,x_d,'r:', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,2), plot(t,y(:,2),'b', t,y_d,'r:', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,3), plot(t,y(:,3),'b', t,z_d,'r:', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,4), plot(t,y(:,4),'b', 'linewidth',4), ylabel('\phi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,5), plot(t,y(:,5),'b', 'linewidth',4), ylabel('\theta (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,6), plot(t,y(:,6),'b', 'linewidth',4), ylabel('\psi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
%-------------------------------------------------------------%
%---3D trajektorija-------------------------------------------%
figure(3)
plot3(y(:,1),y(:,2),y(:,3), 'b', x_d, y_d, z_d, 'r:', 'linewidth',3)
ylabel('x (m)','FontSize',16,'FontName','Times'), xlabel('y (m)','FontSize',
16,'FontName','Times'), zlabel('z (m)','FontSize',16,'FontName','Times'),
% axis([-1.1 1.1 -1.1 1.1 0 20])
set(gca,'fontsize',14,'FontName','Times'),
grid on
axis square
%-------------------------------------------------------------%
%---Sila i momenti--------------------------------------------%
F=diff(y(:,13))./diff(t);
M1=diff(y(:,14))./diff(t);
M2=diff(y(:,15))./diff(t);
M3=diff(y(:,16))./diff(t);
td=t(1:(length(t)-1));
```

```
figure(4)
subplot(2,2,1), plot(td,F,'b', 'linewidth',3), ylabel('F_z (N)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 9 15])
subplot(2,2,2), plot(td,M1,'b', 'linewidth',3), ylabel('\tau_1 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -10 5])
subplot(2,2,3), plot(td,M2,'b', 'linewidth',3), ylabel('\tau_2 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 5])
subplot(2,2,4), plot(td,M3,'b', 'linewidth',3), ylabel('\tau_3 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 1])
%-------------------------------------------------------------%
%---2D projekcije trajektorije--------------------------------%
figure(5)
subplot(2,1,1), plot(y(:,1),y(:,3),'b', x_d, z_d,'r:','linewidth',4), xlabel('x
(m)','FontSize',16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'),
%axis([-1.1 1.1 -1.1 1.1])
subplot(2,1,2), plot(y(:,2),y(:,3),'b', y_d, z_d,'r:', 'linewidth',4), xlabel('y
(m)','FontSize',16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'),
%axis([-1.1 1.1 0 20])
%-------------------------------------------------------------%
end
```

Matlab – upravljanje po brzini

```
% REGULACIJA QUADROTORA PO BRZINAMA
clear all; close all;
clc
T=10; %simulation time
global mm Ixx Iyy Izz id_zz g Vx0 Ay0 k0 k1 k2 k3 Kpz Kdz KK2 KK1 KK0 KP
global Kiz Kpw Kdw ky0 ky1 ky2 ky3 QQ r_g Sg I_B Ax Ay Az Apsi
mm = 1.6;
Ixy = 0; Iyz = 0; Ixz = 0;
Vx0=0.5; Ay0=1;
g=9.81; x_g=0.0; y_g=0.0; z_g=0.0; r_g=[x_g; y_g; z_g];
Ixx = 0.27; Iyy = 0.69; Izz = 0.50; Ixy = 0; Iyz = 0; Ixz = 0; % momenti inercije
I_B = [Ixx -Ixy -Ixz; -Ixy Iyy -Iyz; -Ixz -Iyz Izz];
QQ = 1; % MODEL 1 (full rigid-body dynamic model)
%QQ = 2; % MODEL 2 (simplified rigid-body dynamic model)
%QQ = 3; % MODEL 3 (more simplified rigid-body dynamic model)
KP = 8; % P pojacanje
%% amplitude referentnih step signala brzina:
Ax = 0.5; %(TREBA biti < 0.8 da bi sva tri modela imala slican odziv - pretpostavka
linearizacije)
Ay = 0.5; %(TREBA biti < 0.8
Az = 0.5; %(TREBA biti < 0.8
Apsi = 0.0; %(mora biti mali: <0.1 da bi sva tri modela imala slican odziv -
pretpostavka linearizacije)
%Poja?anja Z, polovi
a1= -3; a2= -5; a3 = -4;
Kpz = a1*a2+a1*a3+a2*a3;
Kdz = -(a1+a2+a3);
Kiz = -a1*a2*a3;
KK2 = Kdz;
KK1 = Kpz;
KK0 = Kiz;
%Poja?anja Psi, polovi
b1= -3; b2 = -2;
Kpw = b1*b2;
Kdw = -(b1+b2);
%Poja?anje x,y
```

```
c1 = -2; c2 = -5; c3 = -3; c4= -4;
k0 = c1*c2*c3*c4;
k1 = -(c1*c2*(c3+c4)+c3*c4*(c1+c2));
k2 = c1*c2+c3*c4+(c1+c2)*(c3+c4);
k3 = -(c1+c2+c3+c4);
ky0 =k0;
ky1 =k1;
ky2 =k2;
ky3 =k3;
xx0=zeros(1,17);
if (QQ == 2)|(QQ == 3)
xx0(7)=0*0.1; xx0(9)=0*0.1; xx0(11)=0*0.1; % initial angles
end
if (QQ == 1)
xx0(4)=0*0.1; xx0(5)=0*0.1; xx0(6)=0*0.1; % initial angles
end
xx0(17) = -0.95*mm*g/Kiz; % pocetno stanje integratora
%options = odeset('RelTol',1e-6,'AbsTol',1e-6);
%[t,y] = ode45('dron1',[0 T],xx0,options);
tspan=[0,T]; Nstep=10000; DelT = T/Nstep;
[t, y] = rk4(@dron1_brzina, tspan, xx0, DelT);
%-----------------------------------------
%---Reference trajectory-------------------------------------%
dx_d = Ax*(0.5*(1+sign(t-2.2)) + 0.5*(1+sign(6-t)) - 1);
dy_d = Ay*(0.5*(1+sign(t-2.2)) + 0.5*(1+sign(6-t)) - 1);
dz_d = Az*0.5*(1+sign(2-t));
dPsi_d = Apsi*(0.5*(1+sign(t-2.2)) + 0.5*(1+sign(5-t)) - 1);
if (QQ == 2)|(QQ == 3)
%------MODEL 2----MODEL 3-----------------------------------%
%-----------------------------------------------------------%
figure(1)
subplot(2,3,1), plot(t,y(:,1),'b', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,2), plot(t,y(:,3),'b', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,3), plot(t,y(:,5),'b', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,4), plot(t,y(:,7),'b', 'linewidth',4), ylabel('\phi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,5), plot(t,y(:,9),'b', 'linewidth',4), ylabel('\theta (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,6), plot(t,y(:,11),'b', 'linewidth',4), ylabel('\psi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
%-----------------------------------------------------------%
%---3D trajektorija------------------------------------------%
figure(2)
plot3(y(:,1),y(:,3),y(:,5), 'b', 'linewidth',3)
ylabel('x (m)','FontSize',16,'FontName','Times'), xlabel('y (m)','FontSize',
16,'FontName','Times'), zlabel('z (m)','FontSize',16,'FontName','Times'),
% axis([-1.1 1.1 -1.1 1.1 0 20])
set(gca,'fontsize',14,'FontName','Times'),
grid on
axis square
%-----------------------------------------------------------%
%---Sila i momenti-------------------------------------------%
F=diff(y(:,13))./diff(t);
M1=diff(y(:,14))./diff(t);
```

```
M2=diff(y(:,15))./diff(t);
M3=diff(y(:,16))./diff(t);
td=t(1:(length(t)-1));
figure(3)
subplot(2,2,1), plot(td,F,'b', 'linewidth',3), ylabel('F_z (N)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 9 15])
subplot(2,2,2), plot(td,M1,'b', 'linewidth',3), ylabel('\tau_1 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -10 5])
subplot(2,2,3), plot(td,M2,'b', 'linewidth',3), ylabel('\tau_2 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 5])
subplot(2,2,4), plot(td,M3,'b', 'linewidth',3), ylabel('\tau_3 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 1])
%-------------------------------------------------------------%
%---2D projekcije trajektorije---------------------------------%
figure(6)
subplot(131), plot(y(:,1),y(:,5),'b', 'linewidth',4), xlabel('x (m)','FontSize',
16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'), %axis([-1.1 1.1
-1.1 1.1])
subplot(132), plot(y(:,3),y(:,5),'b', 'linewidth',4), xlabel('y (m)','FontSize',
16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'), %axis([-1.1 1.1
0 20])
subplot(133), plot(y(:,1),y(:,3),'b', 'linewidth',4), xlabel('x (m)','FontSize',
16,'FontName','Times'), ylabel('y (m)','FontSize',16,'FontName','Times'), %axis([-1.1 1.1
0 20])
%-------------------------------------------------------------%
figure(5)
% subplot(2,3,1), plot(t,y(:,1),'b', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
% subplot(2,3,2), plot(t,y(:,3),'b', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
% subplot(2,3,3), plot(t,y(:,5),'b', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
% subplot(2,3,4), plot(t,y(:,2),'b', t,dx_d,'r:', 'linewidth',4), ylabel('\phi
(rad)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
% subplot(2,3,5), plot(t,y(:,4),'b', t,dy_d,'r:', 'linewidth',4), ylabel('\theta
(rad)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
% subplot(2,3,6), plot(t,y(:,6),'b', t,dz_d,'r:', 'linewidth',4), ylabel('\psi
(rad)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,1), plot(t,y(:,1),'b', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,2), plot(t,y(:,3),'b', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,3), plot(t,y(:,5),'b', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,4), plot(t,y(:,11),'b', 'linewidth',4), ylabel('\psi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,5), plot(t,y(:,2),'b', t,dx_d,'r:', 'linewidth',4), ylabel('x^{(1)}
(m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
```

```
subplot(2,4,6), plot(t,y(:,4),'b', t,dy_d,'r:', 'linewidth',4), ylabel('y^{(1)}
(m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,7), plot(t,y(:,6),'b', t,dz_d,'r:', 'linewidth',4), ylabel('z^{(1)}
(m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,8), plot(t,y(:,12),'b', t,dPsi_d,'r:', 'linewidth',4), ylabel('\psi^{(1)}
(rad/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
%-------------------------------------------------------------%
end
if (QQ == 1)
%------MODEL 1-----------------------------------------------------%
%-------------------------------------------------------------%
figure(1)
subplot(2,3,1), plot(t,y(:,1),'b', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,2), plot(t,y(:,2),'b', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,3), plot(t,y(:,3),'b', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,4), plot(t,y(:,4),'b', 'linewidth',4), ylabel('\phi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,5), plot(t,y(:,5),'b', 'linewidth',4), ylabel('\theta (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,3,6), plot(t,y(:,6),'b', 'linewidth',4), ylabel('\psi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
%-------------------------------------------------------------%
%---3D trajektorija-------------------------------------------%
figure(3)
plot3(y(:,1),y(:,2),y(:,3), 'b', 'linewidth',3)
ylabel('x (m)','FontSize',16,'FontName','Times'), xlabel('y (m)','FontSize',
16,'FontName','Times'), zlabel('z (m)','FontSize',16,'FontName','Times'),
% axis([-1.1 1.1 -1.1 1.1 0 20])
set(gca,'fontsize',14,'FontName','Times'),
grid on
axis square
%-------------------------------------------------------------%
%---Sila i momenti--------------------------------------------%
F=diff(y(:,13))./diff(t);
M1=diff(y(:,14))./diff(t);
M2=diff(y(:,15))./diff(t);
M3=diff(y(:,16))./diff(t);
td=t(1:(length(t)-1));
figure(4)
subplot(2,2,1), plot(td,F,'b', 'linewidth',3), ylabel('F_z (N)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 9 15])
subplot(2,2,2), plot(td,M1,'b', 'linewidth',3), ylabel('\tau_1 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -10 5])
subplot(2,2,3), plot(td,M2,'b', 'linewidth',3), ylabel('\tau_2 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 5])
subplot(2,2,4), plot(td,M3,'b', 'linewidth',3), ylabel('\tau_3 (Nm)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'), %axis([0 5 -1 1])
```

```
%-------------------------------------------------------------%
%---2D projekcije trajektorije-------------------------------%
figure(6)
subplot(131), plot(y(:,1),y(:,3),'b', 'linewidth',4), xlabel('x (m)','FontSize',
16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'), %axis([-1.1 1.1
-1.1 1.1])
subplot(132), plot(y(:,2),y(:,3),'b', 'linewidth',4), xlabel('y (m)','FontSize',
16,'FontName','Times'), ylabel('z (m)','FontSize',16,'FontName','Times'), %axis([-1.1 1.1
0 20])
subplot(133), plot(y(:,1),y(:,2),'b', 'linewidth',4), xlabel('x (m)','FontSize',
16,'FontName','Times'), ylabel('y (m)','FontSize',16,'FontName','Times'), %axis([-1.1 1.1
0 20])
%-------------------------------------------------------------%
figure(5)
subplot(2,4,1), plot(t,y(:,1),'b', 'linewidth',4), ylabel('x (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,2), plot(t,y(:,2),'b', 'linewidth',4), ylabel('y (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,3), plot(t,y(:,3),'b', 'linewidth',4), ylabel('z (m)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,4), plot(t,y(:,6),'b', 'linewidth',4), ylabel('\psi (rad)','FontSize',
16,'FontName','Times'), xlabel('time (sec)','FontSize',16,'FontName','Times'), set
(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,5), plot(t,y(:,7),'b', t,dx_d,'r:', 'linewidth',4), ylabel('x^{(1)}
(m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,6), plot(t,y(:,8),'b', t,dy_d,'r:', 'linewidth',4), ylabel('y^{(1)}
(m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,7), plot(t,y(:,9),'b', t,dz_d,'r:', 'linewidth',4), ylabel('z^{(1)}
(m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
subplot(2,4,8), plot(t,y(:,12),'b', t,dPsi_d,'r:', 'linewidth',4), ylabel('\psi^{(1)}
 (m/s)','FontSize',16,'FontName','Times'), xlabel('time (sec)','FontSize',
16,'FontName','Times'), set(gca,'fontsize',14,'FontName','Times'),
%-------------------------------------------------------------%
end
```